Escuela de Ingeniería y Arquitectura Universidad Zaragoza

Universidad Zaragoza 1542

Trabajo Fin de Máster

Reconocimiento e interpretación de gestos con dispositivo Leap

Javier Tierz López

Director: Carlos Sagüés Blázquiz

Máster en Ingeniería de Sistemas e Informática

Departamento de Informática e Ingeniería de Sistemas

Escuela de Ingeniería y Arquitectura

Universidad de Zaragoza

Diciembre de 2013

Reconocimiento e interpretación de gestos con dispositivo Leap

RESUMEN

Con el objetivo de encontrar una relación más intuitiva entre personas y ordenadores, en los últimos años se han producido grandes avances en el estudio y aplicación de la interacción hombre-máquina (HCI, *Human Computer Interaction*). En esta área se ubican el reconocimiento de voz, las pantallas táctiles de *smartphones* y *tablets* y el reconocimiento gestual, presente para el gran público a raíz de la salida al mercado de varios dispositivos en el campo del entretenimiento.

En este contexto, a principios de 2013 empezó a venderse el dispositivo Leap Motion, que ha supuesto una pequeña revolución en el mundo de la HCI. Es de destacar por su gran precisión, pequeño tamaño y bajo coste. Frente a otros dispositivos, que hacen el *tracking* en un entorno más amplio, de un cuerpo entero, y a una distancia de un metro o algo más, el Leap permite exclusivamente el *tracking* de dedos y manos.

En el presente Trabajo Fin de Máster se realiza un estudio del Leap, analizando las posibilidades de desarrollo que ofrece e implementando gestos que sean sencillos de realizar, pero a la vez precisos, para hacer el sistema robusto. Asimismo, se elabora un vocabulario gestual y se aplica a un caso práctico: una cocina de inducción.

ÍNDICE

1. Introducción

1.1. Descripción del problema y motivación

Desde la salida al mercado del primer ratón, en los años 80, se ha tendido a desarrollar tecnologías donde primase la usabilidad del humano y su interacción cada vez más intuitiva con la máquina, habitualmente el ordenador. En los últimos años, el campo de los videojuegos ha adquirido un gran protagonismo en la creación de interfaces, además del control de entornos multimedia. En este sentido, el reconocimiento de gestos está tomando cada vez más importancia en los procesos de interacción hombre-máquina.

Hasta ahora la inmensa mayoría de los esfuerzos se han dedicado al campo del entretenimiento y el ocio y hay campos como la domótica [16] o la interacción con electrodomésticos donde todavía el reconocimiento de gestos no se ha implantado demasiado. En nuestra opinión es un campo con mucho futuro y que dará más usabilidad y sencillez a la interacción hombre-máquina. En el presente Trabajo Fin de Máster se trabaja en el reconocimiento de gestos, implementando como ejemplo práctico una cocina de inducción.

Para que el usuario al que va destinada una determinada interfaz tenga un aprendizaje rápido y una experiencia positiva, se debe construir una librería de gestos que ha de tener ciertas propiedades como la sencillez y la robustez [17].

1.2. Análisis del estado del arte

Hasta la reciente aparición del Leap [1], los dispositivos más eficaces para detectar y seguir el movimiento de la mano eran los dispositivos de detección equipados en la mano [27]: guantes equipados con sensores para medir la posición de la mano y los ángulos de las articulaciones, la posición de las yemas de los dedos, la dirección hacia la que señala o la fuerza generada en un dedo. Sin embargo, estos dispositivos cuentan con varias desventajas, como su elevado precio, la pérdida de naturalidad en los movimientos de la mano o la compleja calibración y montaje que requieren para obtener medidas precisas.

Por otra parte, se han utilizado métodos que utilizan visión para resolver el problema del *tracking*. Durante muchos años, se ha basado en el análisis y procesado de imágenes en escala

de grises o en espacio de color como RGB. El problema de ello es que se pierde una dimensión al tratarse de una imagen plana y no se conoce la información relativa a la profundidad. Esta información puede recuperarse, pagando el precio de una cierta indeterminación en las medidas, mediante reconstrucción tridimensional basada en emparejamiento de puntos. En este contexto, la aparición de sistemas de luz estructurada supuso un gran avance, utilizando actualmente sistemas de visión y profundidad (RGB-D) [19]. A menudo se han usado en la investigación los dispositivos relacionados con las videoconsolas, como pueden ser el Asus Xtion Pro Live [23] y la Kinect de Microsoft [24]. Ambos sistemas utilizan sensores RGB-D a 30 fps.

Un problema del uso de los sensores RGB-D es que se suelen utilizar métodos de detección de piel basados en información de color, lo que puede dar falsos positivos debidos a variaciones en la iluminación de la estancia, al color de la piel de la persona o al color de la ropa que lleve.

Con este tipo de dispositivos, aparecen una serie de inconvenientes como pueden ser el alto problema dimensional que suponen los más de veinte grados de libertad que tiene la mano y los movimientos rápidos al ejecutarse un gesto dinámico, que en parte han sido resueltos en el Leap. Esto se debe a los drivers que implementa, que le permiten actuar a un *framerate* de hasta 200 fps. En el momento de la realización del presente Trabajo Fin de Máster, el código usado no se ha hecho público todavía [11], sólo el correspondiente a la API.

1.3. Objetivos

El principal objetivo del presente trabajo es investigar las posibilidades que ofrece un dispositivo novedoso como el Leap Motion, con menos de un año de vida, que es el más preciso en la actualidad para reconocimiento de manos y dedos.

Se procede a definir un vocabulario gestual sencillo e intuitivo para que pueda ser usado por el usuario con un proceso de aprendizaje lo más corto posible. El objetivo es llegar a un compromiso sencillez-robustez, intentando primar ambos aspectos por igual.

Por último, se procede a implementar aplicaciones que usen los gestos en un caso práctico: una cocina de inducción, de forma que el usuario ve el proceso de reconocimiento en tiempo real y obtiene un *feedback* visual instantáneo.

Los objetivos concretos son:

- Estudiar el dispositivo y sus posibilidades para el reconocimiento gestual.
- Definir un vocabulario de gestos que pueda ser utilizado con Leap.
- Implementar algoritmos de detección y reconocimiento de gestos.
- Implementar ejemplos de aplicación con detección y reconocimiento de gestos.

2. Reconocimiento de gestos con otros sensores

El reconocimiento de la profundidad de la escena que se quiere estudiar, mediante un dispositivo que capture imágenes, se engloba dentro del campo de la visión por computador. Para obtener la imagen 3D de la escena [20], los métodos empleados se pueden englobar en dos grupos: activos y pasivos. Los activos intervienen sobre la escena, enviando sobre ella una señal: ultrasonidos, láser, infrarrojos... En este grupo se engloban las técnicas basadas en luz estructurada y tiempo de vuelo, mientras que la visión estereoscópica utiliza métodos pasivos. Las técnicas basadas en luz estructurada o tiempo de vuelo, reducen bastante el coste computacional necesario para resolver el problema y logran un compromiso entre precisión y velocidad de procesado, por lo que son las más usadas para aplicaciones en tiempo real y reconocimiento de gestos.

2.1. Visión estereoscópica

Este método toma como referencia el modelo estereoscópico presente en algunos seres vivos y particularmente en los seres humanos, donde el desplazamiento relativo de los ojos en un mismo eje permite obtener la profundidad de los objetos situados en la escena mediante un inconsciente método de triangulación a partir de la imagen generada de la misma escena por cada ojo.

El sistema más habitual de visión estereoscópica (Figura 1) consta de dos cámaras con sus ejes ópticos (Z_I y Z_D) paralelos entre sí y separados una distancia que se denomina línea base. Ambas cámaras tienen sus ejes ópticos perpendiculares a la línea base y las rectas epipolares paralelas a la línea base. Las rectas epipolares son las líneas que proyectan un mismo punto de la escena en la imagen izquierda y derecha. Este sistema de ejes ópticos paralelos implica que las imágenes de un mismo punto captado por las dos cámaras difieren sólo en la componente horizontal y como consecuencia se obtiene la llamada restricción epipolar, que ayuda a encontrar las correspondencias entre ambas imágenes. La disparidad d para cada par de puntos emparejados $P_I(x_I,y_I)$ y $P_D(x_D,y_D)$ viene dada por $d=x_I - x_D$

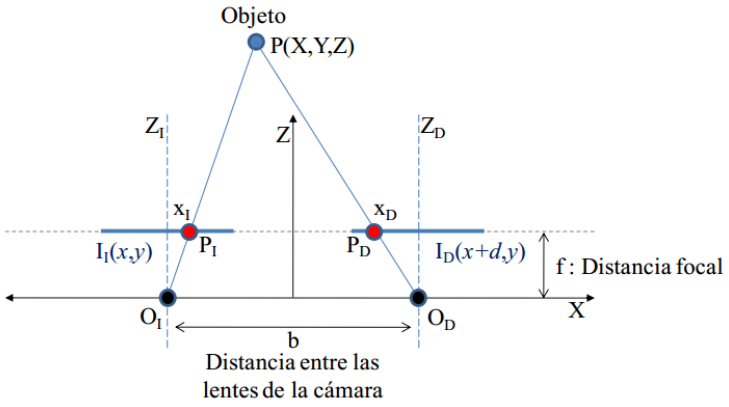

Figura 1. Geometría de visión esteroscópica con dos cámaras

Considerando la semejanza de triángulos, se puede obtener la profundidad del punto P de la escena, como se indica en las siguientes ecuaciones (Figura 2). Se observa que la profundidad Z es inversamente proporcional a la disparidad de la imagen.

$$\left.\begin{aligned} O_I: \frac{\frac{b}{2}+X}{Z} &= \frac{x_I}{f} \\ O_D: -\frac{\frac{b}{2}-X}{Z} &= \frac{x_D}{f} \end{aligned}\right\} \Rightarrow \left.\begin{aligned} x_I &= \frac{f}{Z}\left(X+\frac{b}{2}\right) \\ x_D &= \frac{f}{Z}\left(X-\frac{b}{2}\right) \end{aligned}\right\} \Rightarrow d = x_I - x_D = \frac{fb}{Z} \Rightarrow \boldsymbol{Z = \frac{fb}{d}}$$

Figura 2. Cálculo de la profundidad de un punto 3D

Para establecer la correspondencia entre las imágenes de las dos cámaras, se pueden distinguir dos grupos de técnicas: basadas en el área y basadas en las características. Estas últimas restringen la búsqueda de correspondencias a un conjunto de propiedades obtenidas a partir de descriptores, como los puntos SIFT o los SURF, que procesan y extraen las características de una imagen. En los métodos basados en área, los elementos a comparar son ventanas de la imagen de dimensión fija, y el criterio de semejanza es una medida de la correspondencia entre las ventanas de las dos imágenes. Dentro de este grupo se pueden distinguir métodos como la suma de diferencias absolutas (SAD), la suma de diferencias al cuadrado (SSD), la correlación cruzada normalizada (NCC) y el coeficiente de correlación de Pearson.

En el año 2006, Klaus [29] propone un algoritmo que utiliza la segmentación del color en la imagen de referencia y maximiza el número de correspondencias fiables. Se descompone la imagen en regiones de color homogéneo o escala de grises, asumiendo que los valores en esas regiones varían suavemente y que las discontinuidades sólo ocurren en los límites de cada región. Se modelan un conjunto de planos de disparidad, que determinan la disparidad d para cada píxel de la imagen y se obtiene un mapa de disparidad de la misma. Lo cual implica que se obtiene un mapa de profundidad, ya que ambas variables son inversamente proporcionales. En la Figura 3 se detallan los distintos pasos de este algoritmo.

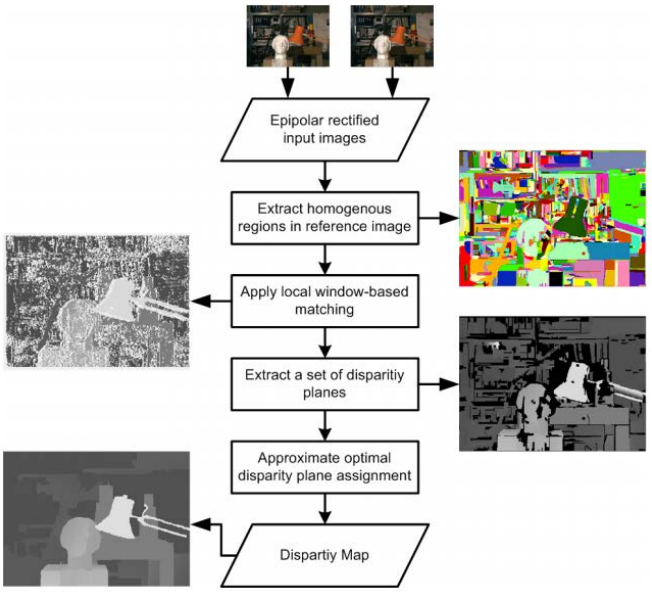

Figura 3. Diagrama del algoritmo de correspondencia estéreo basado en segmentación

2.2. Luz estructurada

Los dispositivos basados en luz estructurada están compuestos por una o más cámaras y una o más fuentes de luz, aunque lo más habitual es una cámara y una fuente de luz. Frente a la estereoscopía pasiva, no requiere un calibrado previo ni usar restricciones epipolares. La fuente de luz proyecta una imagen con una cierta estructura o patrón sobre la escena (Figura 4), de forma que cada píxel en el proyector tiene un código asociado, identificable y diferente del resto, ya sean niveles de gris, color o representaciones geométricas. Existe una relación directa entre dichos códigos y sus coordenadas en el plano del proyector. Se captura la imagen del patrón sobre la escena y se decodifica el patrón, obteniéndose un código para cada píxel de la cámara relacionado con el píxel correspondiente del proyector. Conociendo la relación entre el patrón y el plano del proyector, se resuelven las correspondencias entre la cámara y el proyector.

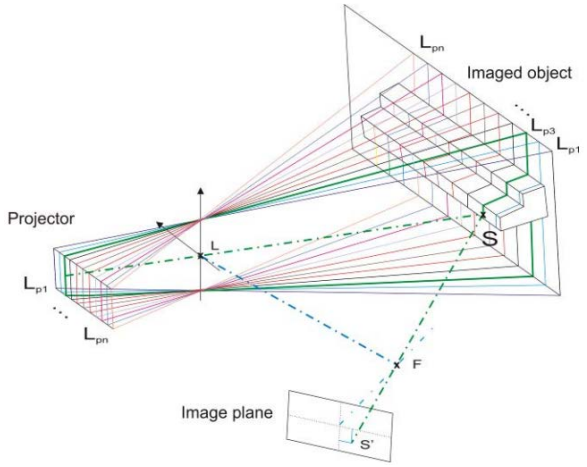

Figura 4. Visión 3D con luz estructurada

En [30] se propone su clasificación en tres grupos: de infrarrojos (IRSL), de luz imperceptible (ISL) y de luz filtrada (FLS).

La luz estructurada infrarroja suele emitirse en el infrarrojo cercano, lo cual permite que la escena pueda ser captada por una cámara CCD y no sea necesaria una cámara de infrarrojos. Esta técnica es usada, entre otros campos, para el escaneo 3D y la navegación de robots.

Por otra parte, los dispositivos basados en ISL están formados por una fuente de luz y dos cámaras. La fuente de luz proyecta sobre la escena un patrón seguido de su complementario (patrón inverso). Una de las cámaras está sincronizada con la proyección del primer patrón y reconstruye la escena con las hallando las correspondencias de la luz estructurada, mientras que la otra cámara tiene un tiempo de integración mayor y observa la escena bajo una luz uniforme debido a la superposición del patrón y el patrón inverso, pudiendo así capturar una imagen en escala de grises o en color y procesarla. Esta técnica permite aunar las ventajas de la luz estructurada y de la visión clásica (color, análisis de texturas…), para conseguir una reconstrucción de la escena 3D más precisa.

La técnica FLS utiliza una fuente de luz acoplada a un filtro de infrarrojos, que actúa como un filtro pasa-baja, anulando las frecuencias más altas que la frecuencia de corte. La principal ventaja de su uso frente a la IRSL es la posibilidad de codificar patrones.

A continuación se observa el patrón de puntos de la Kinect (Figura 5), aparecida en el año 2010, que usa la técnica de luz estructurada.

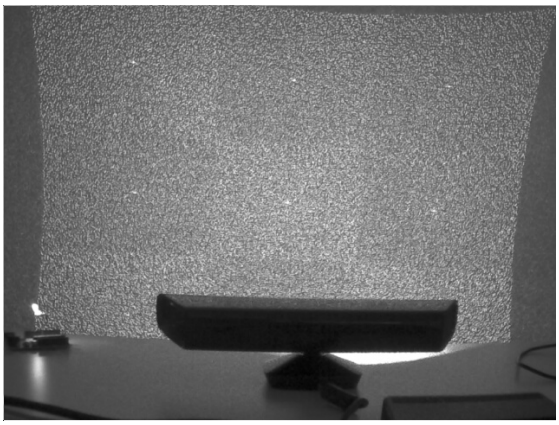

Figura 5. Patrón de puntos de Kinect

El establecimiento de esta correspondencia es uno de los desafíos más importantes de esta técnica y en los últimos años se han propuesto varias estrategias de codificación para los patrones, lo cual afecta sensiblemente al rendimiento de esta técnica. Asimismo, una luz externa con similares características a la luz emisora, puede llegar a arruinar las medidas. Recientemente, se han desarrollado escáneres de luz estructurada que capturan y procesan los datos a 120 fps, más que suficiente para el reconocimiento de expresiones faciales y gestos dinámicos.

2.3. Tiempo de vuelo

Este método halla la distancia a un punto de la escena a partir de la medida directa o indirecta del tiempo que tarda una señal en llegar al punto y volver al dispositivo [22], conocida la velocidad de la luz. Para ello se emite una señal, que en algunos dispositivos es en forma de pulsos y en otros es una onda continua de forma sinusoidal, y se mide el desfase entra la onda emitida y la reflejada después de interaccionar con la escena, como muestra la Figura 6. Este desfase es proporcional a la distancia entre el emisor y la escena.

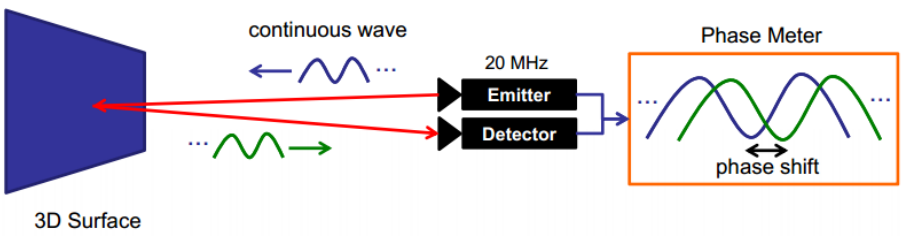

Figura 6. Método de tiempo de vuelo

La correlación cruzada entre ambas señales responde a la siguiente integral, donde τ es el offset:

$$c(\tau) = s * g = \int_{-\infty}^{\infty} s(t) \cdot g(t+\tau)dt$$

Y la distancia buscada es, como se ha indicado anteriormente, proporcional al desfase (ϕ) entre la señal emitida y la recibida:

$$d = \frac{c}{4\pi\omega}\phi$$

Actualmente, los rangos de los dispositivos basados en esta técnica van desde unos centímetros hasta sesenta metros, siendo la resolución de un centímetro o menor. Su *framerate* es elevado, pudiendo alcanzar los 160 fps aunque la resolución suele ser baja, de 320x240.

Una cámara de tiempo de vuelo ilumina la escena LEDs o diodos láser, debido a que la luz ha de ser modulada a frecuencias de 100 MHz o superiores. Una lente focaliza la luz y las imágenes de la escena sobre el sensor imagen, previo paso por un filtro que sólo deja pasar la luz con la misma longitud de onda que el emisor, de forma que se evita la contaminación de los datos por la luz del entorno. La luz focaliza incide sobre el sensor imagen, donde se mide para cada píxel el tiempo que ha transcurrido desde la emisión de luz hasta su llegada al sensor.

Hace unos años, la integración a lo largo del tiempo requería reducir drásticamente el ruido y aun así, los *framerates* que se obtenían y la resolución no eran demasiado elevados, ya que estaban limitados por el tiempo de integración. A menudo aparecía también *motion blur* por esta misma causa. De este modo, no se podían utilizar para aplicaciones en tiempo real. A pesar de ello, gran parte de los problemas han sido resueltos y los sensores de tiempo de vuelo actuales, como el Infineon 3D Image Sensor IRS10x0C [26] o el Camboard Nano [25] pueden funcionar en torno a 90 fps.

Figura 7. Infineon 3D (izquierda) y Camboard Nano (derecha)

3. Análisis del Leap Motion

En el presente apartado se analiza el hardware y el software del Leap Motion, indicándose la tecnología en la que se basa y sus características técnicas. Se incide especialmente en las posibilidades que ofrece para la realización de aplicaciones en tiempo real. Asimismo, se desglosa la información que el Leap permite obtener de dedos y manos en cada *frame*.

3.1. Hardware

El Leap Motion ([1] y [2]) es un pequeño dispositivo de apenas 50 g de peso, con un *framerate* que puede alcanzar los 200 fps y una precisión de 0.01 mm. Se alimenta por USB 2.0 o 3.0 y transfiere datos al driver instalado en el ordenador mediante el puerto USB. Está diseñado para descansar sobre el escritorio de un usuario, mirando hacia arriba, creando así un espacio de interacción 3D de aproximadamente 8 pies cúbicos (≈ 0.23 m^3), siendo éste en forma de pirámide invertida. Dentro de este espacio, el Leap *trackea* manos y dedos, así como objetos cuya geometría sea similar a la de un dedo, tales como bolígrafos y lapiceros, con una precisión muy alta. Esto diferencia el producto de la Kinect, que es más conveniente para el seguimiento de todo el cuerpo en un espacio del tamaño de una sala de estar.

En [4] se muestra el desmontaje de un Leap. Contiene dos pequeñas cámaras y tres LEDs de infrarrojos (Figura 8), que siguen el movimiento de los dedos de una persona con una precisión de una centésima de milímetro. La parte superior es un plástico de color negro que sirve como filtro óptico que sólo transmite luz infrarroja. Cuando el dispositivo detecta una fuente de infrarrojos externa, debida por ejemplo a la iluminación de la estancia, se autocorrige.

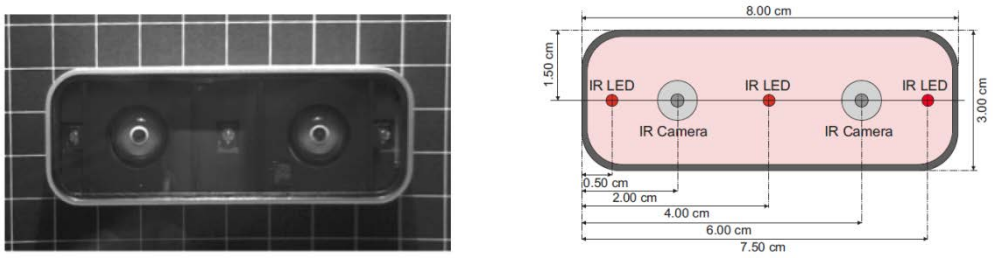

Figura 8. Distrubución de cámeras y LEDs en el Leap Motion

Con el Leap conectado y ejecutando la aplicación de visualizado que viene por defecto, el uso de la CPU sin seguimiento de dedos y manos es del 2%, en un ordenador con procesador i7. Cuando se colocan las manos encima del dispositivo y está enviando datos al driver a través

del USB, el uso de CPU aumenta, pero apenas llega al 10% tras dos minutos de *tracking* continuo de dos manos, por lo que el gasto en CPU es ciertamente bajo teniendo en cuenta las prestaciones que ofrece el Leap. Por otra parte, el consumo de potencia es también reducido, ya que sólo es alimentado por un puerto USB 2.0/3.0. A pleno rendimiento la intensidad eléctrica alcanza 400mA, lo que a 5V implica un consumo máximo de 2W.

En cuanto a la tecnología y los drivers que subyacen a su gran precisión y velocidad de *tracking* poco se puede apuntar de momento. Desde la empresa que lo fabrica no han compartido los detalles concretos de la tecnología empleada, si bien uno de los fundadores ha indicado que, a diferencia de la Kinect, no crea un *grid* de puntos que sean *trackeados* para ver qué se ha movido y hacia dónde. De lo que se deduce que no usa luz estructurada. A pesar de tener dos cámaras, se indicó que tampoco utiliza técnicas de estereovisión para determinar la profundidad. Además, su escaso gasto de CPU también indica que no utiliza la estereovisión pasiva, ya que requeriría más potencia de cálculo. Según esa misma fuente, la segunda cámara sirve para proporcionar una fuente adicional de información y prevenir los errores debidos a las oclusiones de partes de una mano consigo misma o con la otra mano.

En algunas webs de tecnología como [11], se afirma que se basa en tiempo de vuelo. Y parece que más que el propio hardware, son los drivers y el filtrado de información para obtener sólo los datos que interesan para realizar los gestos, los que permiten el potente funcionamiento del Leap Motion. En todo caso, hasta que sus desarrolladores no saquen a la luz algo más de información, todo son especulaciones.

El dispositivo cuesta 80 dólares e incluye dos circuitos relativamente simples, siendo el chip el que se encarga de la conexión USB. Todo el proceso necesario para el seguimiento de gestos se realiza mediante el software del controlador instalado en la computadora del usuario. El hardware no hace demasiado trabajo, cumpliéndose el objetivo planteado por la empresa de utilización del mínimo hardware posible (de ahí su reducido precio). Leap Motion cuenta con un *store* llamado Airspace [12], en el que los desarrolladores van colgando las aplicaciones de escritorio, juegos, etc, si bien de momento no hay demasiada variedad y una parte son de pago. En el blog oficial [13], van apareciendo las novedades respecto a juegos, conferencias y todo lo relacionado con el Leap.

Según los datos ofrecidos por el fabricante, los requisitos mínimos para ejecutar el Leap Motion son los siguientes:

- Windows 7/8, Mac OS X 10.6 o Ubuntu 12.04
- Procesador AMD Phenom II o Intel Core i3/i5/i7
- 2 GB RAM
- USB 2.0
- Conexión a Internet (para acceder al Airspace)

3.2. Software

En Youtube y otras plataformas y en blogs de desarrolladores, van apareciendo constantemente nuevas demostraciones de cómo mediante gestos se puede controlar el software existente como el escritorio de Windows 8 [15] o el Fruit Ninja y crear nuevas aplicaciones para simular un teclado, jugar a los dardos o tocar un instrumento musical virtual. También ha aparecido un plugin para navegar en Chrome con el Leap [14] y se puede usar en Google Earth [28]. Recientemente, la compañía HP ha sacado a la venta el modelo HP Envy 17 [8] con un Leap integrado, ajustado a la geometría del pórtatil. Una aplicación novedosa que está siendo testada en varios hospitales y centros de investigación técnica [9], utiliza el Leap para la telemedicina, interactuando con imágenes médicas en 3D en entornos estériles sin tener contacto físico con ellas. En un futuro no muy lejano, está pensado integrarse en los *smarthpones*, con las consecuencias revolucionarias que ello supondría en nuestra interacción con los móviles.

En [18] se realiza un estudio para comprobar si las prestaciones y la precisión del Leap son realmente las que indica el fabricante, usando un robot industrial. Teóricamente, la precisión del Leap es 0.01 mm, pero en condiciones reales se demuestra que es más impreciso. Para el tracking estático se demuestra que es de 0.2 mm, más que suficiente teniendo en cuenta que la precisión del dedo humano es de 0.4 mm. Para gestos dinámicos es de 0.7 mm, al menos un orden de magnitud menor que la Kinect. En este estudio también se varía el grosor del objeto *trackeado* y los resultados no cambian apreciablemente, por lo que el Leap puede ser usado por personas de toda edad, sexo y fisonomía.

El SDK de Leap Motion permite crear una aplicación con una gran variedad de lenguajes de programación, como C++, C# y Unity, Objetive-C, Java, JavaScript, Python y AS3 [6]. Para el presente trabajo se ha elegido desarrollar en Flex/AS3. Como se ha indicado anteriormente, el Leap detecta y *trackea* manos, dedos y objetos de forma geométrica similar a los dedos, como bolígrafos, informando de sus posiciones 3D, gestos y movimientos. Las distancias vienen dadas en milímetros y las velocidades en milímetros por segundo. El campo de vista toma la forma de una pirámide invertida centrada en el dispositivo, cubriendo un volumen de 0.23 m^3 y el rango efectivo se extiende aproximadamente de 25 a 600 mm por encima del dispositivo. El sistema de coordenadas que utiliza es el cartesiano siguiendo la regla de la mano derecha y centrado en el punto medio del dispositivo (Figura 9).

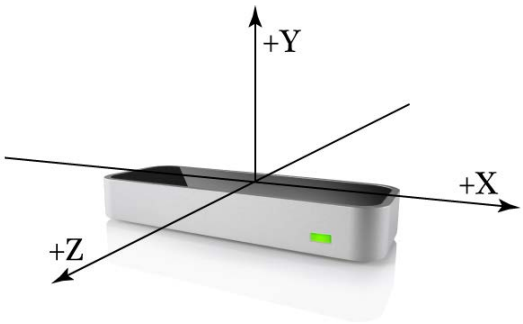

Figura 9. Sistema de coordenadas del Leap

3.2.1. *Frame*

Para cada frame se actualiza un objeto Frame, que contiene toda la información de la escena en ese instante determinado:

- *Hands* - Todas las manos
- *Pointables* – Todos los dedos y objetos similares a un dedo
  - *Fingers* – Todos los dedos
  - *Tools* – Todos los objetos similares a un dedo
- *Gestures* – Todos los gestos que empiezan, terminan o actualizan su estado

En la API, la diferencia entre la asignación *Finger* vs *Tool* es que *Tool* es más largo, más estrecho y más recto, sin dar datos concretos de cómo realiza ese filtrado [5]. En todo caso, en las aplicaciones que se desarrollan en el presente Trabajo Fin de Máster, para obtener los datos y procesarlos, se han elegido *Pointables*, por lo que no se distingue entre *Fingers* y *Tools*.

Esta distinción podría ser importante en el caso de implementar una aplicación en la que se usase un bolígrafo para dibujar, para que el posible ruido de los dedos *trackeados* no intercediese con el *Tool* usado como pincel.

Hay que hacer notar que los *Pointables* se asocian con *Hand*, de forma que se accede a las variables de cada *Pointable* a través de un objeto *Hand.* Sin embargo, también puede ocurrir que haya *pointables* no asociados a ninguna mano, por ser ésta *trackeada* sólo parcialmente en el campo de vista del Leap. A cada *Pointable* o *Hand* detectada se le asigna un identificador único, válido mientras siga siendo *trackeado* ese objeto. Sin embargo, en el momento en el deja de serlo -por haber salido del campo de vista del Leap o por haber sido ocluido por otra mano o dedo- al volver a aparecer se le asigna un nuevo identificador, de valor *random*. Es decir, el ID sólo vale desde el primer frame en que se *trackea* hasta que desaparece. Hay que ser cuidadoso con esto a la hora de desarrollar aplicaciones.

El software del Leap analiza el movimiento desde el *frame* anterior al actual y sintetiza la traslación, rotación y factores de escala. Se analizan todos los objetos presentes en el campo de vista en un determinado frame. De este modo, si se colocan dos manos sobre el Leap, el objeto *Frame* basa las transformaciones realizadas entre ambas manos. Sin embargo, se puede acceder a los atributos de cada *Hand* por separado si así se requiere. Los atributos de cada objeto *Frame*, basados entre el frame actual y el anterior son los siguientes:

- *Eje de rotación*
- *Ángulo de rotación*
- *Matriz de rotación*
- *Factor de escala*
- *Traslación*

3.2.2. *Hands*

Un objeto *Hand* proporciona la información de la posición 3D, características y movimiento de una mano detectada y también de los dedos y objetos asociados a ella. El Leap puede detectar más de dos manos a la vez en el caso de que haya más de una persona con sus manos en el campo de visión, aunque para evitar oclusiones no se recomienda este uso. Para cada mano *trackeada*, se tienen los siguientes atributos:

- Posición del centro de la palma
- Velocidad de la palma
- Normal de la palma
- Dirección de la mano
- Centro de la esfera que forma
- Radio de curvatura

La dirección de la mano y la normal de la palma son vectores unitarios (de módulo 1), que describen la orientación de la mano respecto al sistema de coordenadas del Leap. Por otra parte, el centro de la esfera y su radio de curvatura son parámetros de una esfera cuya superficie encaja con la mano (Figura 10).

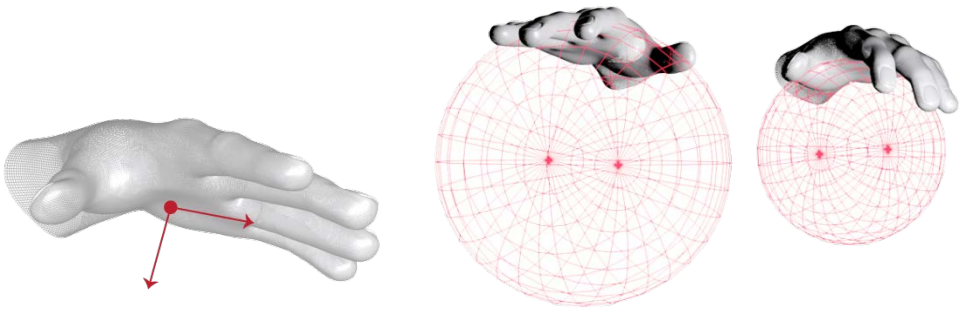

Figura 10. Vector normal a la palma (izquierda) y esfera formada por la mano (derecha, con 2 radios diferentes).

Asimismo, es posible acceder a la información de traslación, rotación y factor de escala, a los que se hace referencia en 3.2.1.

3.2.3. *Pointables*

Como se ha dicho anteriormente, en los objetos *Pointables* se incluyen *Fingers* y *Tools*. Las características físicas que se pueden obtener de cada *Pointable* son las siguientes:

- *Longitud*
- *Anchura*
- *Dirección*
- *Posición del extremo*
- *Velocidad del extremo*

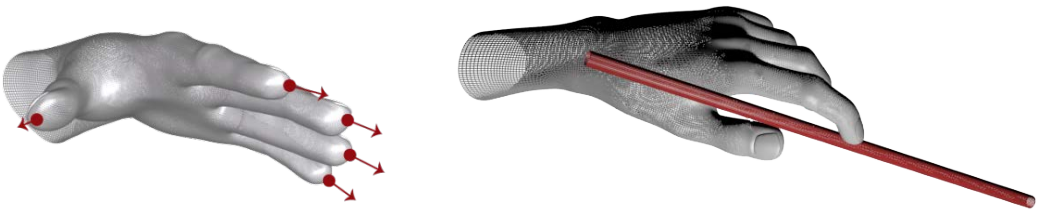

Figura 11. Dirección de los dedos y de un Tool

3.2.4. Gestures

El Leap permite reconocer ciertos patrones de movimientos, que pueden ser accedidos en cada frame mediante el objetivo *Gestures*. Si un gesto determinado continúa existiendo a lo largo de un determinado tiempo, se van actualizando sus atributos en los siguientes frames. Estos son los cuatro gestos catalogados por el Leap:

- *Circle* – Dedo trazando un círculo
- *Swipe* – Barrido lineal de la mano
- *Key Tap* – Movimiento de un dedo similar a presionar una tecla
- *Screen Tap* – Movimiento de un dedo similar a pulsar una pantalla vertical

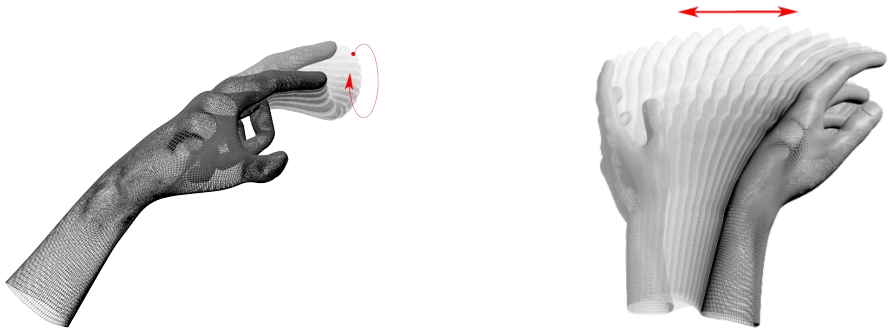

Figura 12. Gestos Circle (izquierda) y Swipe (derecha)

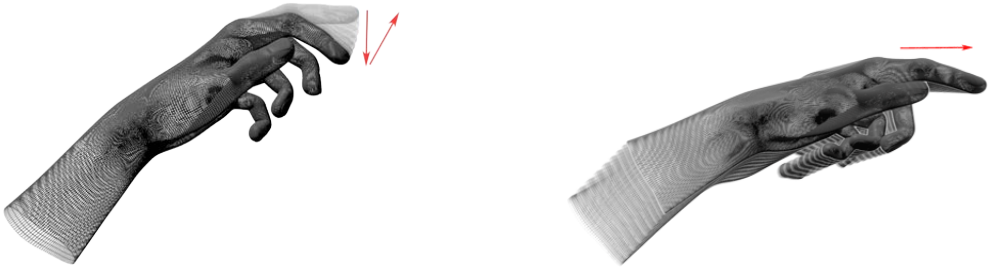

Figura 13. Gestos Key Tap (izquierda) y Screen Tap (derecha)

Los gestos *Circle* y *Swipe* son continuos (duran varios frames), mientras que *Key Tap* y *Screen Tap* son discretos (duran un frame). Los dos primeros actualizan su estado hasta que el gesto termina: en el caso de *Circle* cuando el dedo se sale de la geometría del círculo o la velocidad de trazado es muy baja y en el de *Swipe* cuando se mueva demasiado despacio.

Debido a la precisión del Leap se pueden detectar varios gestos por segundo, pero la mayoría de ellos son superfluos para la aplicación que se desarrolla. Por ejemplo, al mover levemente la mano en una dirección, el Leap permite reconocer un Swipe que no se ha realizado conscientemente o varios, concatenados. El filtrado de los gestos según las características que se requieren es fundamental si se quiere dotar de una gran robustez a nuestra aplicación e impedir la aparición de falsos positivos.

4. Reconocimiento y filtrado de gestos con Leap Motion

En este apartado se presenta el vocabulario de gestos que se ha propuesto en el presente trabajo. Se realiza un estudio a cinco personas que no han usado nunca el Leap y se analizan los resultados obtenidos, justificando los parámetros usados para la elección del filtrado de gestos. Por último, se implementa el filtrado y se analiza la robustez del mismo, realizando los gestos una persona con experiencia en el uso del Leap.

4.1. Introducción

Como se ha comentado en el apartado anterior, el Leap reconoce a menudo gestos que no se han realizado conscientemente, por la gran precisión que tiene. Por ello, es preciso filtrar adecuadamente los datos para el correcto reconocimiento de gestos y que las acciones que llevan aparejadas no se realicen si el gesto no se corresponde con el que se ha definido. Es decir, para que la aplicación sea robusta, evitando en lo posible los falsos positivos, se ha de filtrar llegando a un compromiso robustez vs usabilidad, favoreciendo en todo caso la robustez. Esto se debe a que es preferible que se haya de repetir el gesto alguna vez a que se realice una acción sin haberlo deseado, ya que dañaría seriamente la experiencia del usuario al emplear la aplicación. En todo caso, al empezar a usar cualquier dispositivo que es nuevo para una persona, aparece una curva de aprendizaje y se tarda un cierto tiempo en manejarlo con corrección, por lo que no creemos que esto represente un inconveniente, primando como se ha dicho la robustez. Tómese como ejemplo el uso de un ratón para una persona que no lo ha utilizado nunca: su aprendizaje no es instantáneo, y requerirá algo de tiempo hasta que lo sepa emplear con cierta soltura. La curva de aprendizaje deseable implica que se alcance al máximo de efectividad -mínimo porcentaje de falsos positivos- en el menor número de pruebas posible. Los gestos implementados en el presente trabajo son los siguientes:

- Tracking de un dedo durante un periodo de tiempo
- Elección del radio de un círculo
- Cursor
- Selección mediante profundidad
- Selección mediante Screen Tap
- Barrido horizontal izquierdo
- Barrido horizontal derecho
- Barrido vertical hacia abajo
- Pulgar hacia la izquierda
- Pulgar hacia la derecha

Para el filtrado de gestos, se hace un estudio a cinco personas que no han utilizado nunca el Leap. El estudio se limita a los gestos de barridos y pulgares. Se indica a las cinco personas el gesto que han de realizar, sin hacer ninguna consideración adicional de la posición u orientación de la mano o dedos. Suponemos que la distribución de datos corresponde a una distribución normal, con la siguiente función de densidad:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

Se halla la media (μ) y la desviación típica (σ) de los datos obtenidos para cada gesto y los resultados se implementan para realizar el filtrado.

4.2. *Tracking* de un dedo durante un periodo de tiempo

Se realiza introduciendo un solo dedo en la escena y lleva asociado un círculo de progreso que se va llenando desde que empieza el gesto hasta que se termina. Permite obtener un *feedback* de que el gesto está efectivamente en proceso de ejecución. Se ha elegido una duración de dos segundos, para dar cierta robustez sin que el tiempo sea demasiado elevado.

4.3. Elección del radio de un círculo

Se usa el radio de la esfera que forma la palma de la mano (Figura 10, derecha) y se representa gráficamente en la interfaz con un círculo sólido de radio proporcional al del radio de la esfera de la palma. Se observa empíricamente que el radio de la esfera que proporciona la API del Leap va entre los 56 mm y los 110 mm, aproximadamente. Para evitar el ruido y que el círculo gráfico oscile en su radio, se ha acotado el mínimo y el máximo a 65 mm y 90 mm, respectivamente. A continuación, se cambia de escala para que el rango en píxeles del círculo gráfico sea el elegido.

4.4. Cursor

Sirve para seleccionar objetos, usando nuestro dedo como un ratón moviéndose en dos dimensiones por la pantalla. Se puede utilizar el método *intersecPointable* de la clase *Screen*, que devuelve un vector normalizado en 3 dimensiones. Para que el puntero pueda ir por toda la interfaz de nuestra aplicación y que un movimiento del dedo en el eje Y positivo se corresponda con subir el puntero, se ha de multiplicar la coordenada X del vector por la anchura en píxeles de la aplicación y la coordenada Y por uno menos la altura en píxeles. Esto se debe a que el pixel (0,0) está en la esquina superior izquierda y si no el movimiento en el eje Y estaría invertido.

4.5. Selección

Se implementan dos gestos alternativos para la acción de seleccionar: mediante profundidad y mediante el gesto Screen Tap (Figura 13, derecha). El primero es más sencillo de realizar para un usuario que no ha utilizado nunca el Leap, mientras que el segundo es más robusto y produce menos falsos positivos.

4.5.1. Selección mediante profundidad

Se utiliza la coordenada Z del extremo del dedo que se utiliza como cursor (4.3). Se elige que la acción de selección se produzca cuando la coordenada Z del cursor es menor que -70 mm (hacia dentro, ver Figura 9). Su funcionamiento es aceptable, aunque su robustez es mejorable y se ha de tener presente la posición del Leap respecto al dedo del cursor, ya que si no puede ocurrir que se esté seleccionando continuamente (si se tiene el dedo muy hacia adentro) o que por el contrario se deba adelantar su posición mucho si se tiene el dedo muy hacia afuera.

4.5.2 Selección mediante Screen Tap

La mínima velocidad por defecto para este gesto es 50.0 mm/s y se decide eliminar esta restricción, con objeto de simplificar su ejecución. Por otra parte, para dotarle de robustez, la componente Z mínima de la dirección se establece en -0.8, es decir, perpendicular al Leap y hacia adentro.

4.6. Barrido

Se han implementado tres gestos de barrido: horizontal hacia la izquierda, horizontal hacia la derecha y vertical hacia abajo. El filtrado de los barridos se realiza en dos pasos. Por una parte, se impone que el tiempo pasado entre el anterior barrido y el actual sea al menos de 500 ms para evitar que se superpongan dos barridos consecutivos. Por otra parte, se eligen unos parámetros de filtrado para que el gesto sea aceptado como correcto.

Se realiza un estudio de cinco personas, repitiendo cada uno de los barridos en diez ocasiones. Se han seleccionado los siguientes parámetros para su análisis:

- Componentes del vector dirección del barrido
- Componentes del vector dirección de la mano
- Posición x de inicio (para el barrido horizontal)
- Posición y de inicio (para el barrido vertical)
- Velocidad

En la figura 14 se indica la dirección de barrido y el ángulo α respecto al eje X en el caso del barrido horizontal y respecto al eje Y en el caso del barrido vertical. Asimismo, se indica la posición de inicio del barrido (Δx y Δy) con respecto al origen de coordenadas, centrado en el Leap.

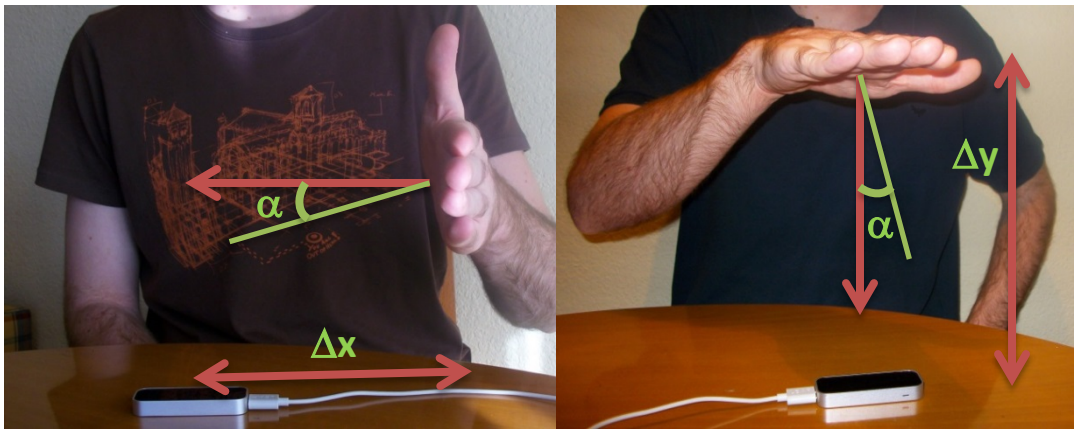

Figura 14. Direcciones de barrido horizontal y vertical y posiciones de inicio

4.6.1. Barrido horizontal izquierdo

Se indica a las cinco personas del estudio que han de pasar la mano horizontalmente por encima del Leap, de derecha a izquierda, repitiendo la acción cada individuo en diez ocasiones. Se toman los datos de los parámetros a los que se ha hecho referencia, obteniéndose los siguientes resultados.

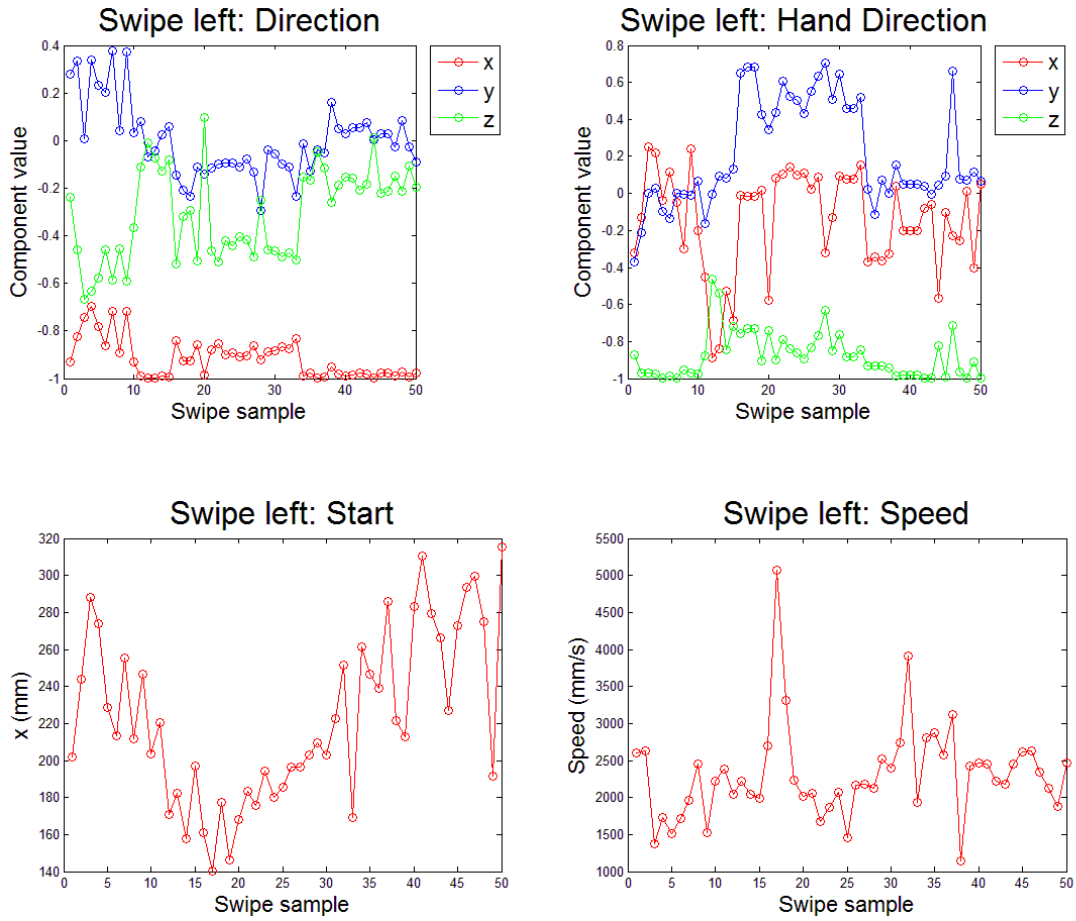

Figura 15. Barrido horizontal izquierdo: componentes de la dirección del barrido (superior izquierda), componentes de dirección de la mano (superior derecha), posición X de inicio (inferior izquierda) y velocidad del barrido (inferior derecha)

En la anterior figura se observa que la dirección del barrido está bastante bien delimitada (gráfica superior izquierda), siendo su componente X en casi todos los casos menor que -0.7. En cuanto a la dirección de la mano (gráfica superior derecha) su componente dominante es la Z, si bien no se ha de ser demasiado restrictivo con este parámetro si queremos obtener una tasa de positivos elevada. Por otra parte, en las gráficas de la coordenada X del inicio y de la

velocidad del barrido (gráficas inferior derecha e izquierda, respectivamente), se aprecia una varianza más alta que en las otras dos gráficas.

La media y la desviación típica de cada una de las distribuciones es la siguiente:

- Dirección X del barrido: μ = -0.916, σ = 0.082
- Dirección Z de la mano: μ = -0.875, σ = 0.123
- Posición X inicial (mm): μ = 222.8, σ = 45.8
- Velocidad (mm/s): μ = 2312, σ = 637

Se observa en la gráfica de la dirección del barrido (Figura 15, gráfica superior izquierda) que las componentes Y y Z son ciertamente aleatorias. El vector dirección del barrido (*swipe*) es unitario, por lo que:

$$swipe_x^2 + swipe_y^2 + swipe_z^2 = 1 \qquad (1)$$

Suponiendo que las componentes Y y Z tienen igual peso, la relación entre las componentes X e Y es, usando (1):

$$swipe_y = \pm\sqrt{\frac{1-swipe_x^2}{2}} \qquad (2)$$

El ángulo α que forma el barrido con el eje de coordenadas X (Figura 14) es, por trigonometría:

$$\alpha = \arctan\left(\frac{swipe_y}{swipe_x}\right) \qquad (3)$$

A partir de (2) y (3), tomando como valor la media obtenida, es decir $swipe_x$ = -0.916, se halla el ángulo α máximo que puede formar el barrido con el eje horizontal para que sea válido.

$\alpha_{máx}$ = 17.2°

4.6.2. Barrido horizontal derecho

En la siguiente fase del estudio se analiza el gesto *Barrido horizontal derecho* realizado por las cinco personas, tomando diez muestras de cada una y obteniéndose los resultados mostrados en la siguiente figura.

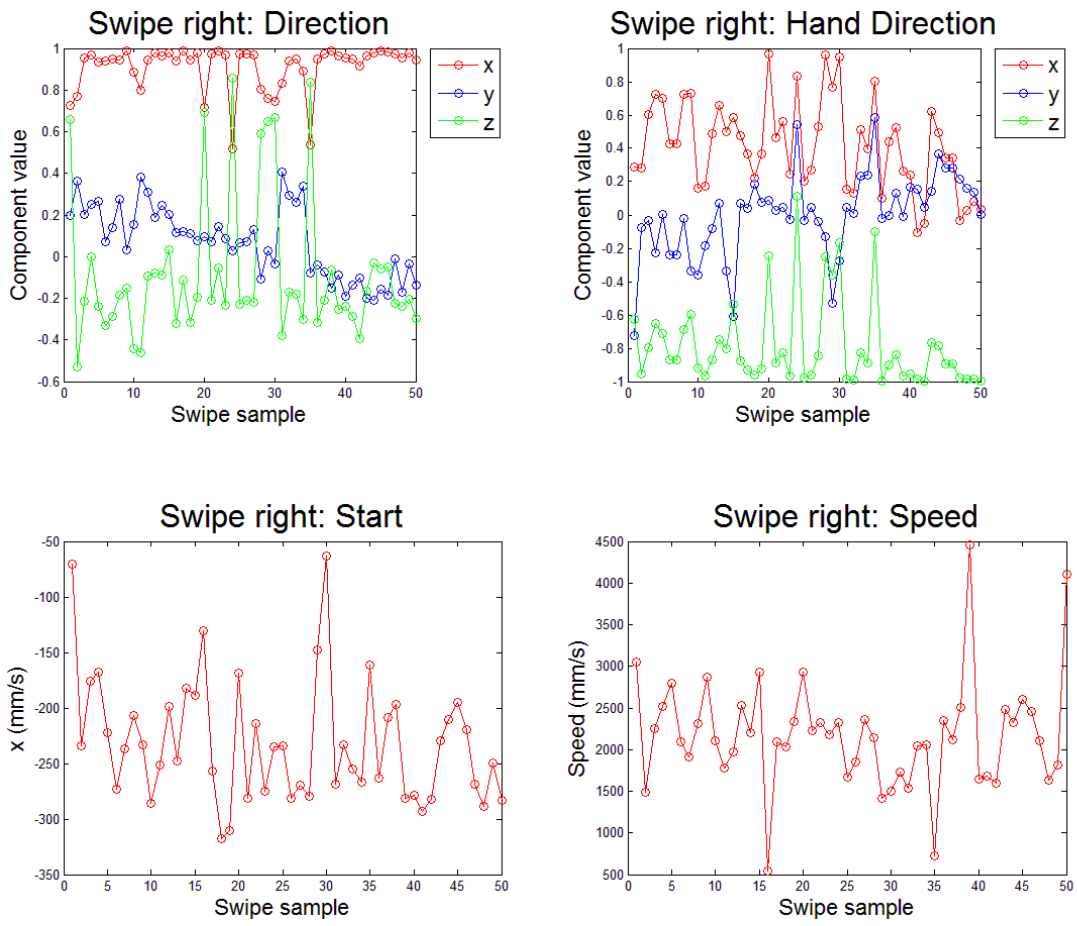

Figura 16. Barrido horizontal derecho: componentes de la dirección del barrido (superior izquierda), componentes de la dirección de la mano (superior derecha), posición X de inicio (inferior izquierda) y velocidad del barrido (inferior derecha)

Para las distintas características de este gesto se obtienen los siguientes resultados.

- Dirección X del barrido: $\mu = 0.911$, $\sigma = 0.109$
- Dirección Z de la mano: $\mu = -0.788$, $\sigma = 0.262$
- Posición X inicial (mm): $\mu = -231.5$, $\sigma = 55.5$
- Velocidad (mm/s): $\mu = 2175$, $\sigma = 661$

Se observa la misma tendencia que en el barrido horizontal izquierdo: la componente X del barrido tiene menos varianza que la posición inicial y la velocidad, mientras que la componente Z de la dirección de la mano es la predominante en la mayoría de las muestras, si bien no tiene un valor acotado.

Utilizando $swipe_x = 0.911$ en (3) se obtiene:

$\alpha_{máx}$ = 17.7°

4.6.3. Barrido vertical hacia abajo

El tercer gesto que se analiza en el estudio es *Barrido vertical hacia abajo*, tomándose diez muestras para cada persona y obteniéndose los resultados que se muestran en la Figura 17.

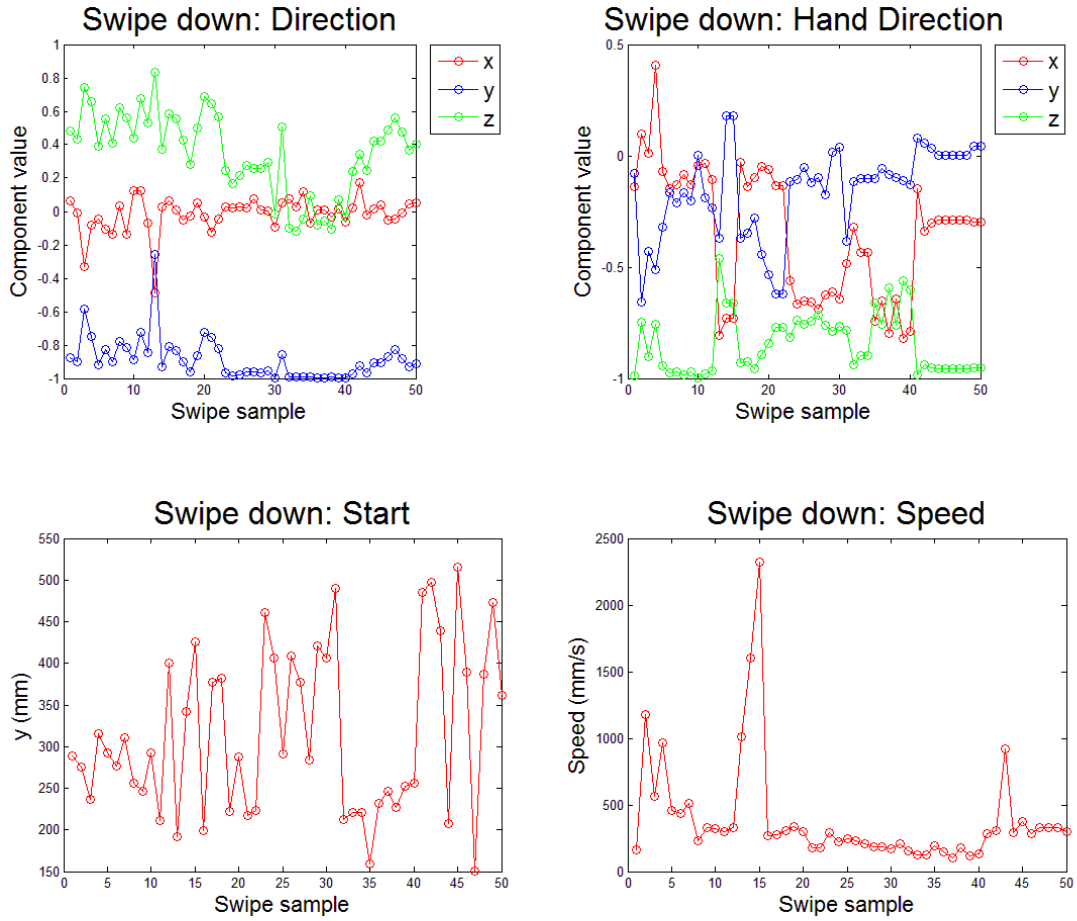

Figura 17. Barrido vertical hacia abajo: componentes de la dirección del barrido (superior izquierda), componentes de la dirección de la mano (superior derecha), posición Y de inicio (inferior izquierda) y velocidad del barrido (inferior derecha)

En la gráfica superior izquierda de la Figura 17 se aprecia que la componente Y del barrido es la predominante, siendo su valor absoluto mayor que 0.7 en casi todas las muestras. Los valores de la posición inicial y la velocidad del barrido vertical muestran una gran varianza, al igual que

ocurre con los barridos horizontales. Además, la velocidad de barrido es mucho menor que en los barridos horizontales, por lo que no se ha de ser restrictivo en el filtrado de ese parámetro.

En las distribuciones de datos de los parámetros, se obtienen los siguientes valores.

- Componente Y de la dirección de barrido: μ = -0.887, σ = 0.128
- Componente Z de dirección de la mano: μ = -0.843, σ = 0.134
- Posición Y inicial (mm): μ = 315.0, σ = 99.2
- Velocidad (mm/s): μ = 390, σ = 404

Para el barrido vertical, al estar el eje de dirección de barrido el Y (Figura 14, derecha), y suponiendo que las componentes X y Z de la dirección del barrido tienen igual peso, las ecuaciones (2) y (3), se reescriben como (4) y (5), respectivamente.

$$swipe_x = \pm\sqrt{\frac{1-swipe_y^2}{2}} \qquad (4)$$

$$\alpha = \arctan\left(\frac{swipe_x}{swipe_y}\right) \qquad (5)$$

El ángulo máximo α que forma el barrido con el eje Y, a partir de $swipe_y$ = -0.887, es:

$\alpha_{máx}$ = 20.2 °

4.7. Pulgar

Se observa en la práctica que, debido a la gran precisión del Leap, aunque se tenga un dedo en la escena, ya sea el pulgar u otro, el sistema detecta en algunos frames dos dedos, dando así lugar a un cierto ruido en las medidas. Con objeto de permitir este ruido, para que este gesto sea válido se admiten uno o dos dedos *trackeados*, aunque se exige que haya una sola mano. Con dos o más manos presentes en la escena, este gesto no es válido.

Si bien el algoritmo implementado no reconoce si se trata del dedo pulgar u otro, con las restricciones geométricas se impone que así sea, ya que la normal a la palma de la mano ha de

tener como componente principal la componente Y, de forma que la mano deba estar horizontal sobre el Leap y el dedo ha de tener su componente principal la X. Hay que señalar que aunque se permita que haya dos dedos en la escena, uno de ellos deberá cumplir todas las restricciones geométricas mínimas que se imponen para que el gesto sea válido.

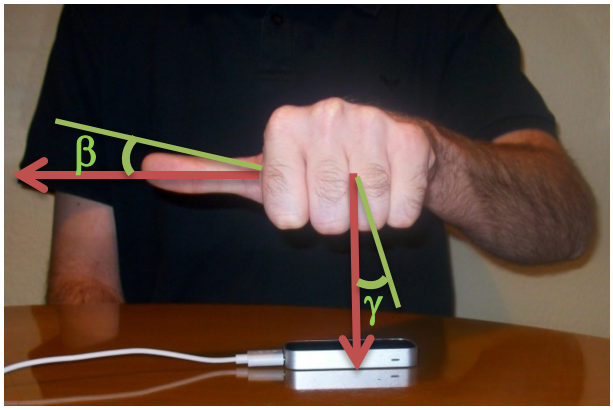

Figura 18. Gesto Pulgar hacia la derecha

En este caso se ha primado la usabilidad frente a la robustez, facilitando que el usuario no tenga que forzar para cerrar el puño, sino poner el pulgar relajadamente apuntando a la derecha o a la izquierda. Se decide que la componente Y de la normal de la palma pueda ser positiva o negativa, siempre que su módulo sea mayor a un cierto valor. De este modo, se permite que el gesto *Pulgar hacia la izquierda* o *Pulgar hacia la derecha* pueda ser realizado con la misma mano. Nótese que mano izquierda pulgar hacia la derecha tiene la componente Y de la normal de la palma negativa (hacia abajo) y mano izquierda pulgar hacia la izquierda tiene la normal positiva (hacia arriba).

Se hace un estudio de cinco personas, igual que en los barridos del apartado 4.5, para el análisis de un conjunto de parámetros. Cada individuo ha de mantener el pulgar fijo y se guardan los datos de cincuenta frames válidos, para el gesto *Pulgar hacia la izquierda* y cincuenta para el *Pulgar hacia la derecha*. Se analizan los siguientes parámetros:

- Componente X del dedo
- Componente Y de la normal a la palma
- Componente Z de la dirección de la mano
- Frames totales
- Frames positivos
- Frames positivos con doble dedo

Se hace un prefiltrado de forma que los frames positivos tienen X como componente principal del vector de la dirección del dedo. Al ser éste un vector unitario, a partir de (1) se obtiene que la componente principal ha de tener un valor

$$Thumb_x \geq \sqrt{\frac{1}{3}} \approx 0.58$$

4.7.1. Pulgar hacia la izquierda

Los resultados obtenidos para el gesto *Pulgar hacia la izquierda* son los representados en la Figura 19, donde sólo se muestran los frames positivos, con componente X del pulgar ≥ 0.58.

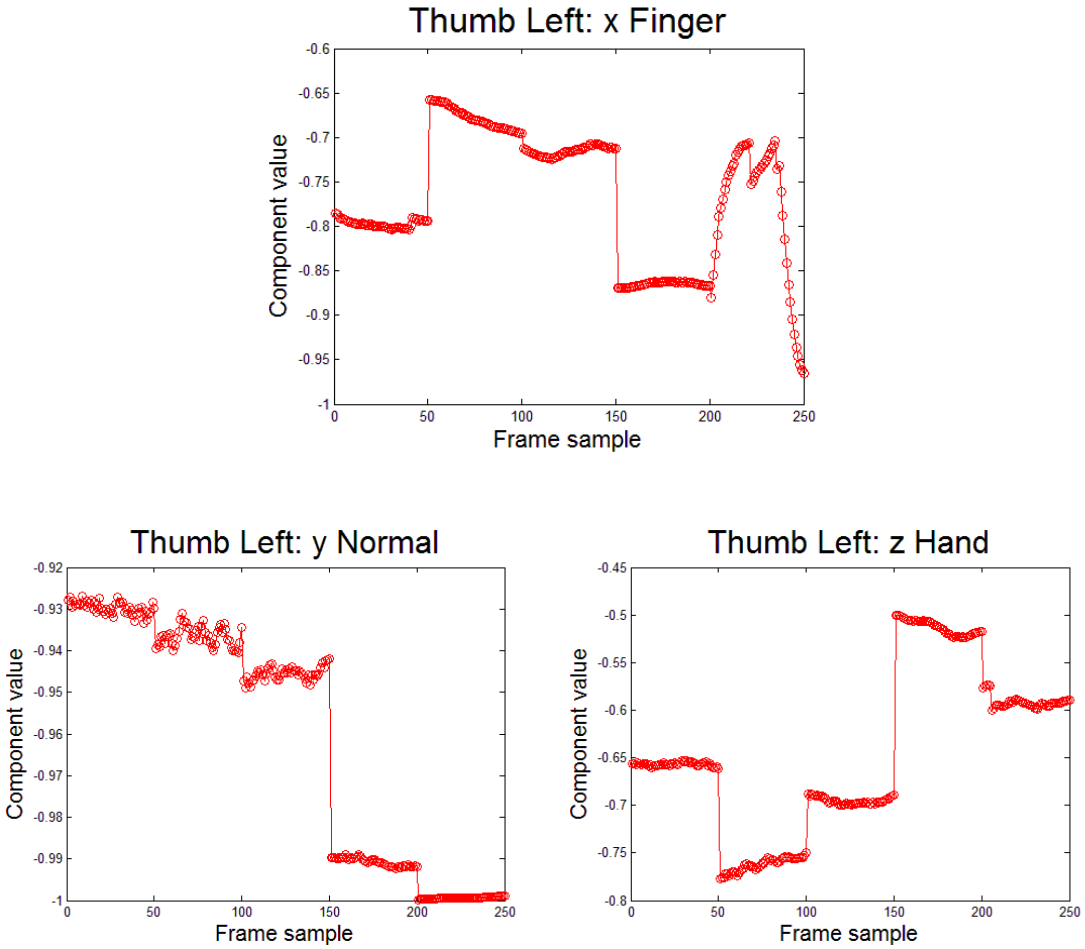

Figura 19. Pulgar hacia la izquierda: componente X del dedo (superior), componente Y de la normal a la mano (inferior izquierda) y dirección Z de la mano (inferior derecha)

Los resultados para la componente Y de la normal de la mano son óptimos y muy acotados (gráfica inferior izquierda). Por el contrario, se observa una varianza mayor en la dirección X del dedo y especialmente en la componente Z de la dirección de la mano. Se decide desestimar el filtrado en dicha componente Z, ya que supondría cierta dificultad para reconocer el gesto como válido y se primará la dirección X del dedo en el reconocimiento tanto de este gesto como del *Pulgar hacia la derecha*. El hecho de que la componente Z de la mano no sea la predominante en muchos de los frames mientras que la X del dedo sí lo sea, puede explicarse fisiológicamente, ya que se ha de forzar la mano para que el pulgar forme un ángulo recto con la mano. Evidentemente, esto puede variar de unos individuos a otros y se ha de tener en cuenta que en el presente trabajo sólo se ha realizado el estudio de cinco individuos.

A continuación se detallan la media y la varianza para los distintos parámetros, así como las tasas de positivos y de detección de dos dedos.

- Dirección X del dedo: $\mu = -0.767$, $\sigma = 0.07$
- Dirección Z de la mano: $\mu = -0.643$, $\sigma = 0.086$
- Dirección Y de la normal: $\mu = -0.960$, $\sigma = 0.029$
- Tasa de positivos: 85.7%
- Tasa de dedos dobles: 14.6%

Para hallar los ángulos β y γ máximos (Figura 18), se utilizan las fórmulas (2) y (3) para β y (4) y (5) para γ. A partir de los valores hallados: $finger_x = -0.767$ y $hand_y = -0.960$, se obtienen los siguientes valores.

$\beta_{máx} = 30.6°$ **$\gamma_{máx} = 11.7°$**

Se observa que el valor de β es muy elevado, permitiéndose un amplio rango de ángulos en la componente X del dedo. Esto se debe en parte a la inexperiencia con el Leap de las personas del estudio y también a la propia fisiología y a la rigidez en la mano que ejerce cada individuo en la realización del gesto.

4.7.2. Pulgar hacia la derecha

Se toman los datos de cincuenta frames para cada individuo realizando este gesto, como se hace en 4.7.1. A continuación se presentan las gráficas con los resultados positivos obtenidos.

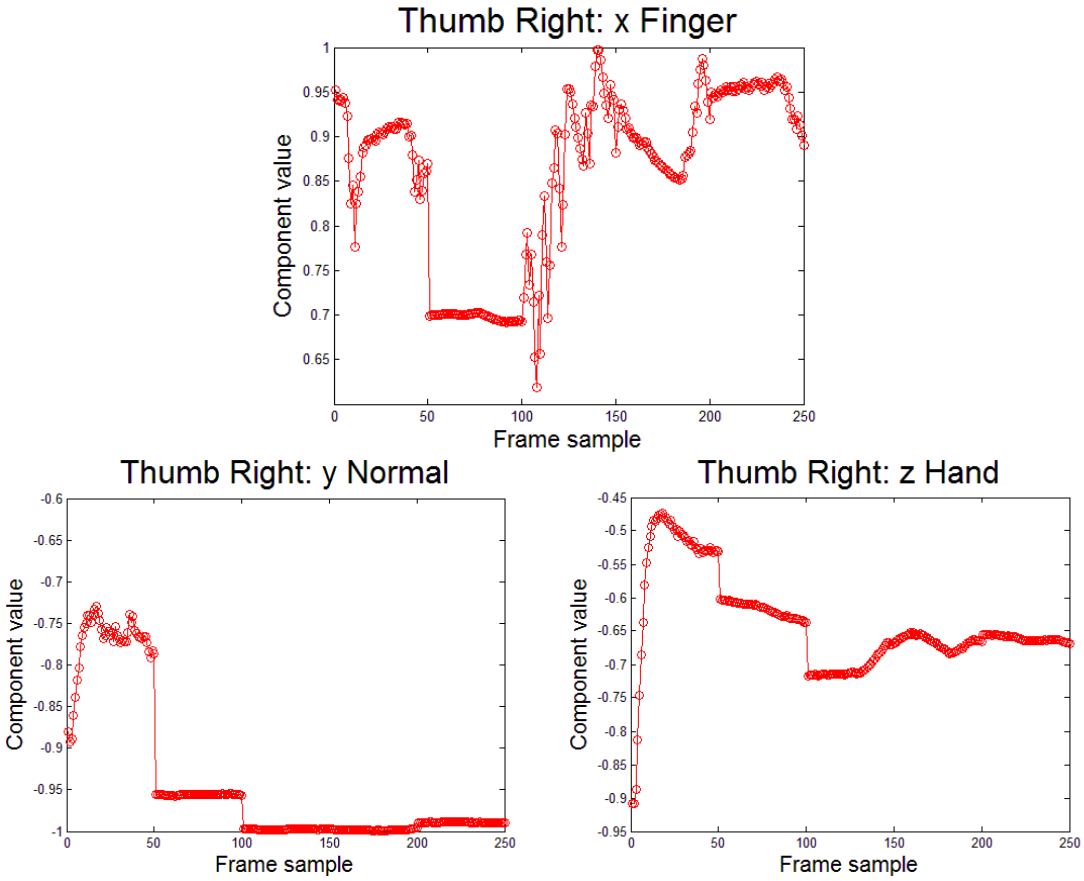

Figura 20. Pulgar hacia la izquierda: componente X del dedo (superior), componente Y de la normal a la mano (inferior izquierda) y dirección Z de la mano (inferior derecha)

Los resultados de la media y la desviación típica para los distintos parámetros de este gesto son las siguientes.

- Dirección X del dedo: $\mu = 0.859$, $\sigma = 0.099$
- Dirección Z de la mano: $\mu = -0.640$, $\sigma = 0.072$
- Dirección Y de la normal: $\mu = -0.942$, $\sigma = 0.088$
- Tasa de positivos: 89.4%
- Tasa de dedos dobles: 13.1%

Los ángulos de desviación máximos son, usando $finger_x = 0.859$ y $hand_y = -0.942$, respectivamente:

$\beta_{máx}$ = 22.9° **$\gamma_{máx}$ = 14.1°**

El valor del ángulo β permitido es elevado, por las razones comentadas en 4.7.1.

4.8. Filtrado y test de uso

Una vez obtenidos y analizados los resultados del estudio a las cinco personas, se eligen los parámetros de filtrado de los gestos de barridos y pulgares y se realiza un test de fiabilidad de los gestos implementados. En este caso, la persona que realiza los gestos ha utilizado el Leap, por lo que tiene más precisión en la realización de los gestos. Se realizan cien pruebas de cada uno de los tres barridos y se toman los valores de *Pulgar hacia la derecha* y del *Pulgar hacia la izquierda* en bloques de cincuenta frames, hasta un total de quinientos frames para cada gesto.

4.8.1 Barridos

Se descarta introducir un filtro de la dirección de la mano, ya que no es un criterio útil de filtrado (ver Figuras 15, 16 y 17, gráfica superior derecha). Asimismo, para el barrido vertical no se tiene en cuenta la velocidad, ya que sus valores (Figura 17, inferior derecha) son muy bajos en comparación con los barridos horizontales. En la Tabla 1 se muestran los umbrales de filtrado que se han introducido para los gestos de Barrido derecho, izquierdo y abajo. Se indican las componentes X e Y de la dirección de barrido, la posición de comienzo del gesto, la velocidad mínima en la ejecución del mismo y el ángulo α máximo de desviación aceptado (ver Figura 14).

	$swipe_x$	$swipe_y$	Δx(mm)	Δy(mm)	Speed(mm/s)	$\alpha_{máx}$(°)
Swipe Left	-0.8	-----	100	-----	1000	28.0
Swipe Right	0.8	-----	-100	-----	1000	28.0
Swipe Down	-----	-0.9	-----	150	-----	18.9

Tabla 1. Umbrales de filtrado para los gestos de Barrido: componentes X e Y de la dirección del barrido, posición X e Y de inicio, velocidad y ángulo máximo de desviación

Una vez introducidos los filtros, se realiza cada barrido en cien ocasiones. En la Figura 21 se presentan las direcciones de los barridos en los tres gestos implementados, donde se puede apreciar la mejora en los resultados obtenidos respecto al apartado 4.7 (Figuras 15, 16 y 17, gráfica superior izquierda), debido a la experiencia del usuario en el uso del Leap.

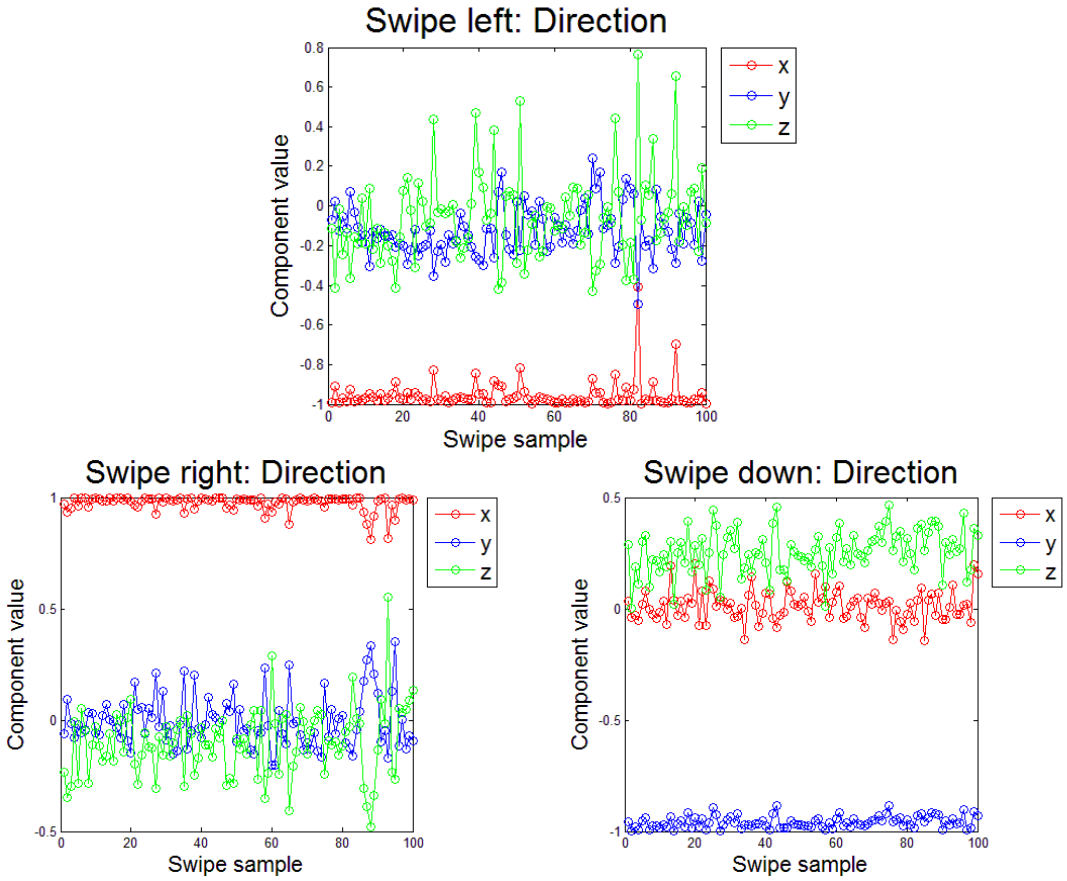

Figura 21. Componentes de las direcciones de los barridos izquierdo, derecho y abajo

Se ha obtenido una tasa de positivos del 100% en el caso de barrido hacia la derecha, del 95% en el barrido hacia la izquierda y del 97% en el barrido hacia abajo. Los valores de la media y la desviación típica -supuesta distribución normal- de las componente X e Y del barrido, de la posición de inicio y de la velocidad son los indicados en la Tabla 2.

	$swipe_x$		$swipe_y$		Δx(mm)		Δy(mm)		Speed(mm/s)	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Swipe Left	-0.954	0.071	-----	-----	234.0	61.5	-----	-----	2057	269
Swipe Right	0.976	0.035	-----	-----	-238.6	42.1	-----	-----	1877	285
Swipe Down	-----	-----	-0.958	0.027	-----	-----	363.6	82.2	291	195

Tabla 2. Media y desviación típica de los parámetros de los gestos Barrido

4.8.2 Pulgares

Se descarta introducir un filtrado respecto a la dirección de la mano, ya que no es un parámetro relevante (Figuras 19 y 20, gráfica inferior derecha) y se filtran la componente X del dedo y la componente Y de la normal a la palma de la mano. Como se ha comentado anteriormente, se permite que se detecten dos dedos, aunque uno de ellos deberá cumplir los parámetros mínimos del filtrado para que el gesto se interprete como correcto. Ambos gestos pueden ser realizados con ambas manos, por lo que se permite que la componente Y de la normal a la palma tenga valor positivo o negativo, siempre que su valor absoluto sea mayor que el umbral, establecido en 0.9.

En la Tabla 3 se muestran la componente X mínima del dedo, la componente Y de la normal a la palma y los ángulos máximos de desviación permitidos (Figura 18).

	$finger_x$	$palm_y$	$\beta_{máx}(°)$	$\gamma_{máx}(°)$
Thumb Left	-0.8	±0.9	28.0	18.9
Thumb Right	0.8	±0.9	28.0	18.9

Tabla 3. Umbrales de filtrado para los gestos de Barrido: componentes X del dedo, componente Y de la normal a la mano y ángulos máximos de desviación

Se toman los datos de cincuenta frames válidos, con diez repeticiones para cada uno de los dos gestos y se obtienen los resultados indicados en la Figura 22.

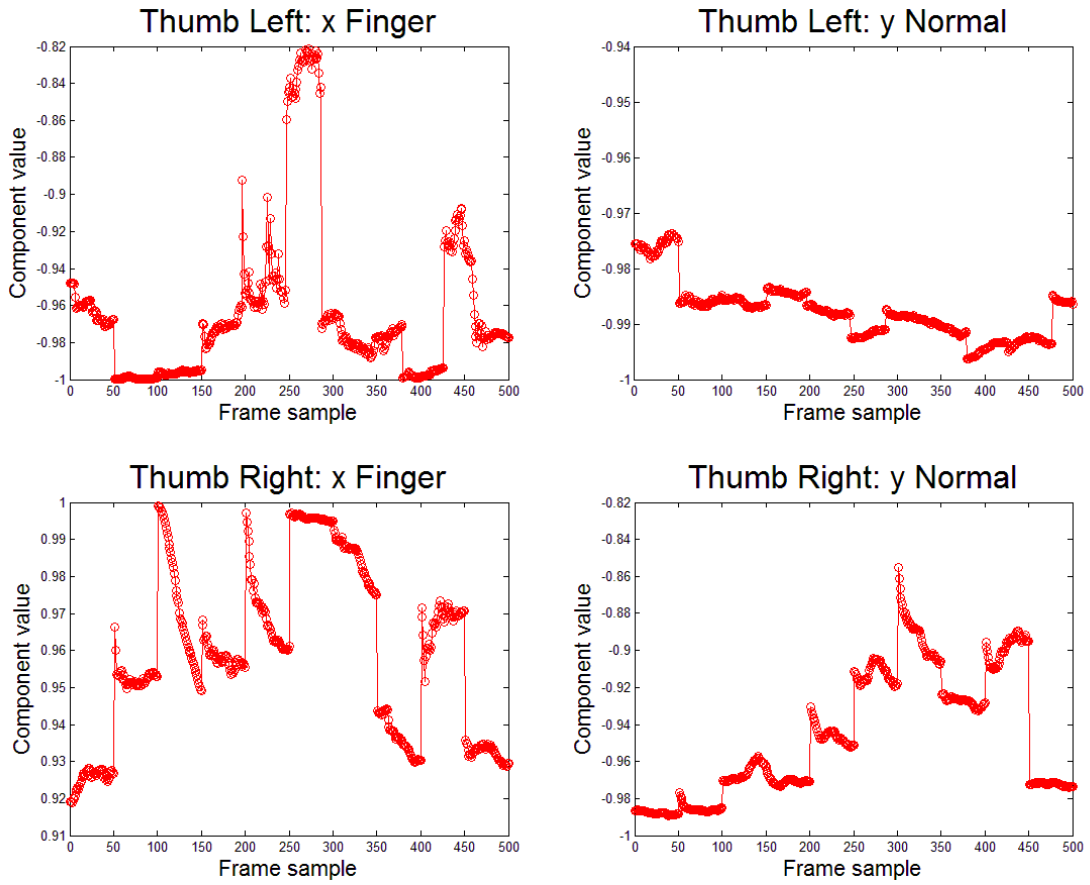

Figura 22. Componente X del pulgar y componente Y de la normal a la palma para el gesto Pulgar hacia izquierda (gráficas superiores) y Gesto Pulgar hacia la derecha (gráficas inferiores)

La tasa de positivos para el gesto *Pulgar hacia la izquierda* es 84.2% y para el *Pulgar hacia la derecha* 89.8%. Los frames inválidos se deben a que el dedo o la mano no son reconocidos o que sus parámetros no cumplen algún parámetro del filtrado que se ha implementado. Por otra parte, la tasa de dedos dobles han sido 15.2% y 12.4%, respectivamente. Los valores de la media y la desviación típica, son los siguientes.

	$finger_x$		$palm_y$	
	μ	**σ**	**μ**	**σ**
Thumb Left	-0.962	0.045	-0.988	0.005
Thumb Right	0.960	0.023	-0.946	0.034

Tabla 4. Media y desviación típica de los parámetros de los gestos Pulgar

5. Aplicaciones implementadas

Para el presente trabajo se han desarrollado las aplicaciones en el lenguaje Flex (mxml + ActionScript 3), usando el Flash Builder 4.7 y la API de Leap Motion para AS3 [6]. En el caso del Visualizador (5.1), se ha usado además la librería Away 3D [7] para crear la escena 3D, la ubicación de la cámara y la iluminación. En este capítulo se exponen las interfaces gráficas de las distintas aplicaciones diseñadas y los gestos elegidos para las transiciones entre estados en cada una de ellas.

5.1. Visualizador

Se trata de una aplicación donde aparece la representación de los dedos en 3D, con los ejes de coordenadas de la mano representados en el centro de la misma. A su derecha, figuran una serie de datos referentes a los objetos *trackeados* en el frame *actual*.

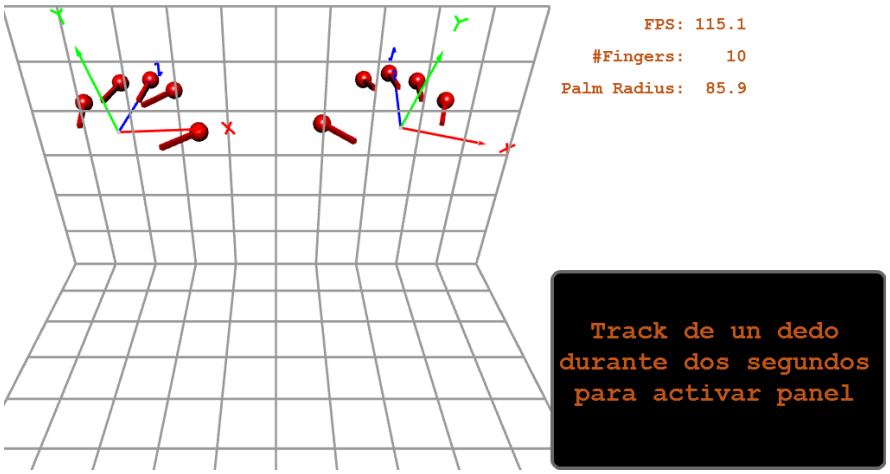

Figura 23. Interfaz de la aplicación Visualizador

En la imagen anterior se observa un *grid* y dentro de él dos manos con todos sus dedos *trackeados*. A la derecha arriba se ve la información de los frames por segundo, el número de dedos y el radio de la esfera que forma la palma de la mano. En este caso, de la primera mano que apareció en la escena, pero en realidad este dato no es relevante hasta que no se active el panel de abajo a la derecha, que representa una cocina de inducción. Para activarlo, se realiza

el gesto indicado en la sección 4.2: *Tracking de un dedo durante un periodo de tiempo*. En el momento que hay sólo un dedo en el campo de visión del Leap, se activa el círculo de *feedback*.

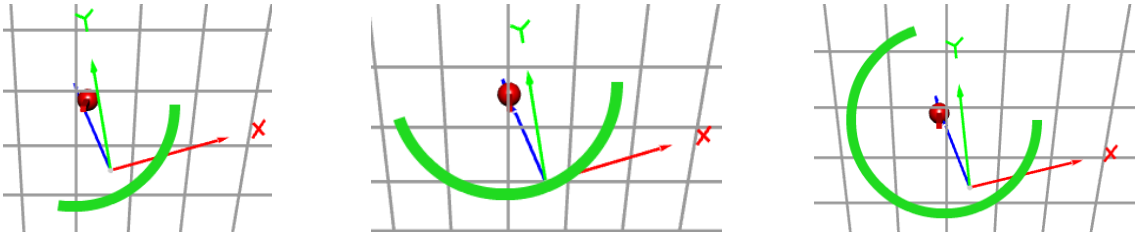

Figura 24. Círculo de progreso del gesto Tracking de un dedo durante un periodo de tiempo

Cuando se ha completado el proceso, el panel se activa y se permite seleccionar el radio de inducción. En la Figura 16 se observa la relación entre el radio de la mano detectada ("Palm Radius").

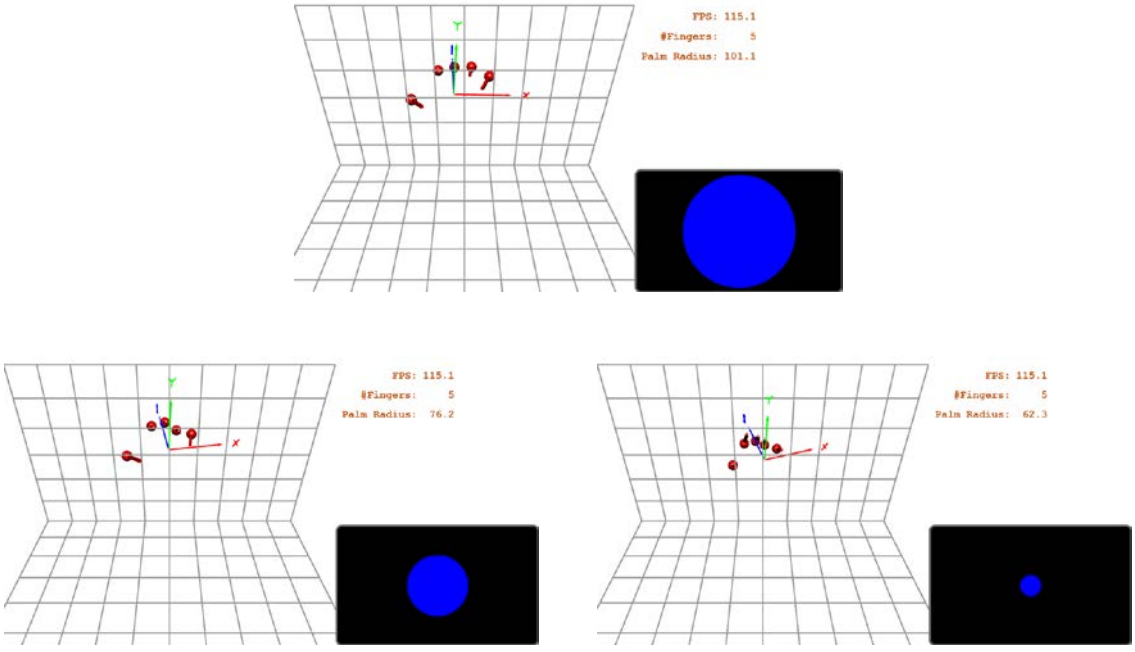

Figura 25. Distintos radios del gesto ***Elección del radio de un círculo***

5.2. Inducción con geometría variable

La interfaz de esta aplicación (Figura 26) consta de una placa de inducción negra a pantalla completa, un cursor verde que aparece cuando se *trackea* un dedo y un rectángulo de selección que se usa para entrar en el estado de modificación del radio del fuego de inducción, mediante el gesto detallado en el apartado 4.2: *Tracking* de un dedo durante un periodo de tiempo. En realidad es una pequeña modificación de éste, ya que es suficiente con que el cursor esté encima del rectángulo, por lo que puede haber más de un dedo *trackeado*. De todos modos, es más sencillo el control del cursor con un solo dedo, habitualmente el índice.

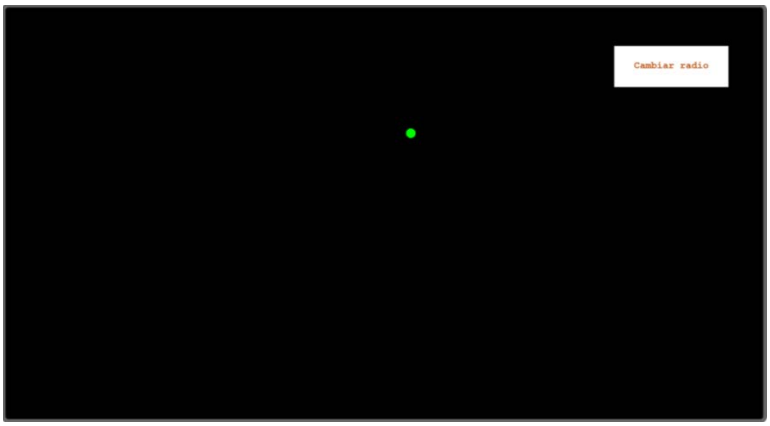

Figura 26. Interfaz de la aplicación Inducción con geometría variable con un círculo verde representando la posición del cursor

A continuación se detalla el pseudocódigo que se implementa en esta aplicación (Figura 27), con sus posibles estados y los gestos necesarios para las transiciones de uno a otro.

```
for each frame
    if cursor is on “button” & target selected
        set progressCircle on;
        if progressCircle is completed
            get palmRadius;
            change graphicRadius;
            if hands = 2
                stove ← fixed;
end
```

Figura 27. Pseudocódigo de la aplicación Inducción con geometría variable

Cuando se realiza el gesto de *Seleccionar* mediante el método explicado en 4.5.1, es decir, colocando el extremo del dedo al menos 70 mm hacia adentro del eje Z del Leap, se fija un círculo de color azul en la coordenada (x,y) de la pantalla en la que se tenga el cursor en ese momento. Representa la posición del fuego que se quiere añadir a la placa de inducción. Para confirmar la posición del fuego, se coloca el cursor durante dos segundos sobre la superficie ("botón") que aparece en la interfaz arriba a la derecha (Figura 28).

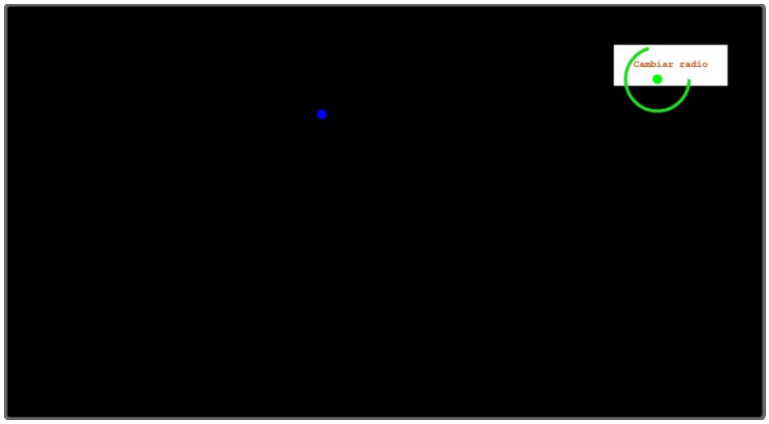

Figura 28. Fuego seleccionado (en azul) y círculo de progreso

Una vez ha acabado el círculo de progreso y la aplicación ha confirmado la posición del fuego, se activa la modificación del radio, mediante el gesto 4.3, usando el radio de la esfera que forma la palma de la mano (Figura 10, derecha).

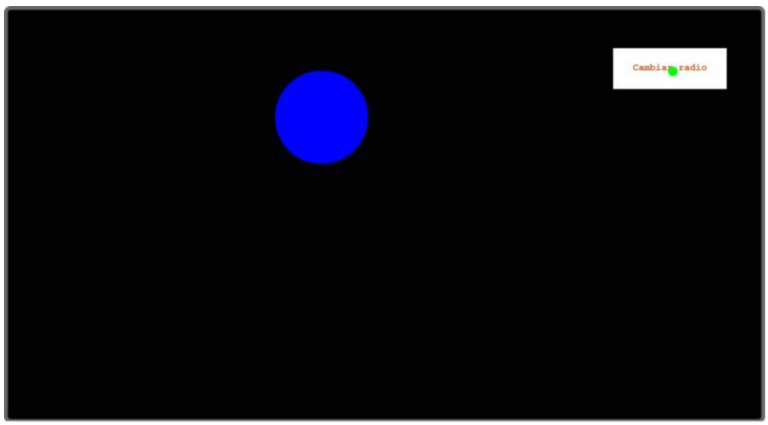

Figura 29. Modificación del radio del fuego

Para confirmar el radio del fuego de inducción, se coloca una segunda mano dentro del campo de visión del Leap. De este modo, el radio que tenga el círculo en ese momento se convierte en rojo, fijándose el radio y el cursor verde se vuelve a activar, permitiendo añadir nuevos fuegos. En la Figura 30 se observa la colocación de tres fuegos de distinto radio, usando repetidas veces el método que se ha explicado.

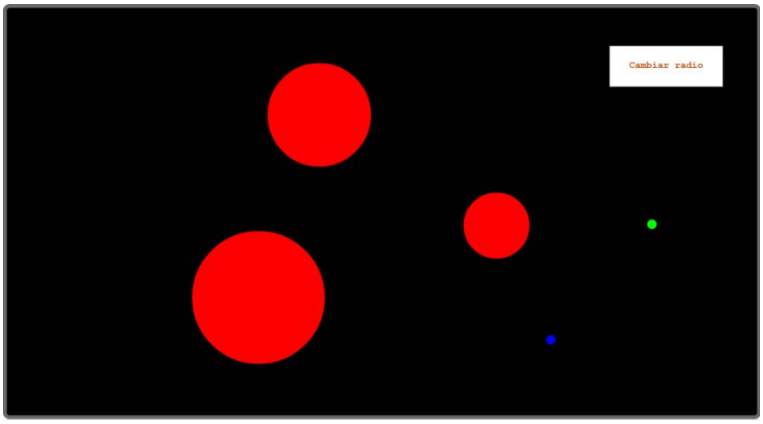

Figura 30. Cocina de inducción con tres fuegos de distinto radio

5.3. Inducción con geometría fija

En esta aplicación se diseña una interfaz gráfica con cuatro fuegos de inducción, con un código de colores que indica en qué estado se encuentra cada uno de ellos, además de un panel en el centro que muestra la potencia a la que están. Inicialmente, el sistema está desconectado y para conectarlo se realiza el gesto de barrido vertical hacia abajo (Figura 14, derecha). A continuación se detalla en pseudocódigo el funcionamiento de la aplicación.

```
for each frame
    if gesture = barridoVertical
        set ON ↔ OFF;
    if state = ON
        set cursorActive true;
        if gesture = screenTap & cursor is over a stove
            set stoveSelected true;
        if any stove is selected
            if gesture = pulgarDerecha
                temperature++;
            if gesture = pulgarIzquierda
                temperature--;
            if gesture = barridoDerecha
                acceptTemperature;
            if gesture = barridoDerecha
                cancelTemperature;
end
```

Figura 31. Pseudocódigo de la aplicación Inducción con geometría fija

En la Figura 32 se muestra la interfaz de la aplicación cuando la cocina de inducción está desactivada y todos los fuegos apagados.

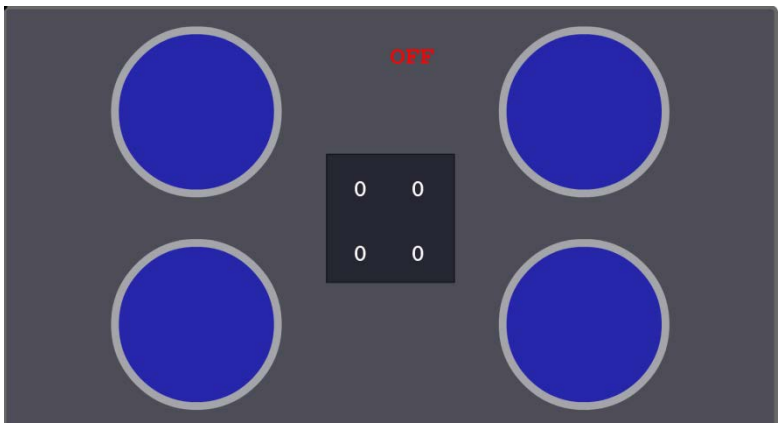

Figura 32. Interfaz de la aplicación Inducción con geometría fija con la cocina desactivada y los fuegos apagados

En la anterior figura, se ven representados los cuatro fuegos circulares, fijos y de igual radio, con sus respectivos marcadores de potencia. Cada uno de los fuegos está formado por dos

círculos concéntricos, que cambian de color en función del estado en el que se encuentren. El círculo externo tiene dos posibles estados: blanco (sin seleccionar) y verde (seleccionado). El círculo interno tiene asimismo dos posibles estados: azul (potencia nula) y rojo (potencia positiva). Los marcadores de potencia del centro de la interfaz también cambian de color en función del estado: blanco (potencia nula), verde (seleccionado) y rojo (potencia confirmada positiva). De este modo se obtiene un *feedback* más completo de la situación. En la Figura 33 la cocina está activa, tras haber realizado el gesto de *Barrido vertical hacia abajo*, pasando el indicador de OFF a ON. En este estado aparece un cursor en la pantalla, que se mueve con un dedo presente en el campo de vista del Leap.

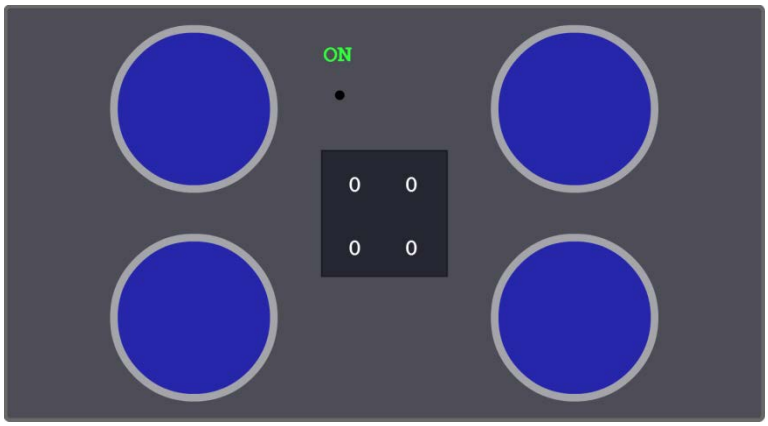

Figura 33. Cocina de inducción activa, con cursor activo en forma de círculo negro

Colocando el cursor sobre uno de los cuatro fuegos y haciendo el gesto de selección señalado en el apartado 4.5.2 (Screen Tap, ver Figura 13 derecha), el fuego se selecciona, quedando su círculo externo marcado en verde y el cursor se desactiva (Figura 34).

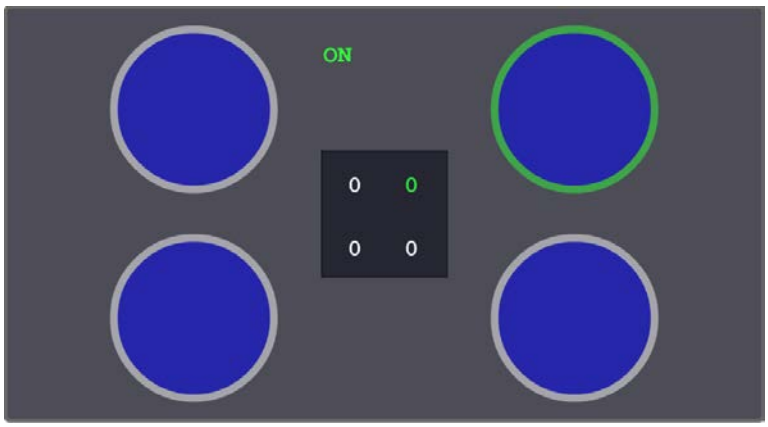

Figura 34. Fuego seleccionado, potencia 0

En este estado, hay cuatro gestos posibles. Por una parte se puede subir y bajar la potencia, con el gesto del *Pulgar hacia la derecha* o *Pulgar hacia la izquierda*, respectivamente (Figura 35). Por otra, se puede aceptar o cancelar la potencia que se ha fijado, mediante *Barrido horizontal derecho* o *Barrido horizontal izquierdo, respectivamente*.

Figura 35. Fuego seleccionado, potencia 7

Si se cancela la potencia seleccionada mediante *Barrido horizontal izquierdo*, el fuego se deselecciona y se descarta la potencia en caso de que se hubiera elegido alguna, quedando de nuevo el círculo externo en blanco y el indicador en cero (Figura 33). Si por el contrario se realiza el gesto *Barrido horizontal derecho*, el fuego que estaba seleccionado adquiere la potencia que se ha introducido, quedando el círculo interno en rojo y el externo en blanco (Figura 36). En cualquiera de los dos casos, se deselecciona el fuego y se activa el cursor, pudiendo así seleccionar uno de los cuatro fuegos.

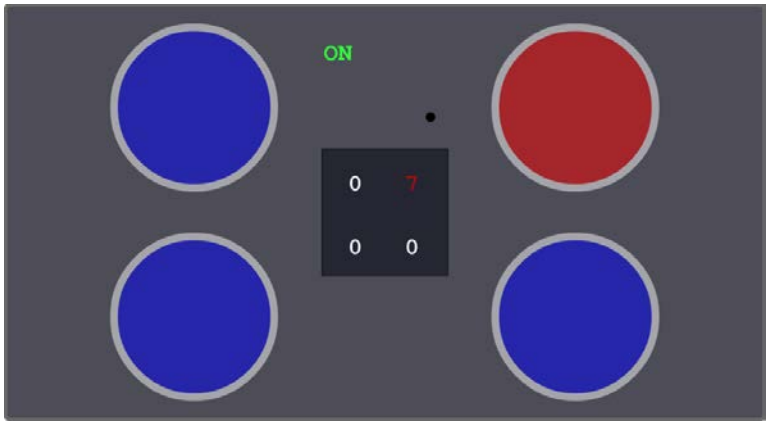

Figura 36. Estado cursor, con fuego encendido a potencia 7

6. Conclusiones

En el presente trabajo se han analizado las posibilidades del Leap Motion para el reconocimiento de gestos y el desarrollo de interfaces de usuario en el campo de la interacción hombre-máquina. Este dispositivo ha salido al mercado en 2013 y destaca por su gran precisión, pequeño tamaño y bajo coste, permitiendo el reconocimiento de dedos y manos. Se ha diseñado un vocabulario de gestos lo más sencillo y preciso posible, para que las aplicaciones sean robustas.

Se ha realizado un estudio a cinco personas que no habían utilizado nunca el dispositivo, analizando los valores de los parámetros más relevantes. Tras este estudio, se ha implementado un filtrado de gestos y se ha demostrado que una persona que ha usado previamente el Leap obtiene una alta tasa de positivos para todos los gestos propuestos, cumpliéndose un compromiso entre usabilidad y robustez.

Una vez diseñados y reconocidos los gestos e implementado su filtrado, se han desarrollado tres aplicaciones: una de visualizado de los dedos en 3D y las otras dos relacionadas con la domótica y más concretamente con las cocinas de inducción. La transición entre los distintos estados de cada aplicación se realiza mediante la ejecución de los distintos gestos que se han diseñado.

Mientras se realizaba este proyecto han ido surgiendo ideas y posibles mejoras en algunos apartados como el perfeccionamiento y ampliación del vocabulario gestual propuesto, buscando tasas de positivos mayores en usuarios sin contacto anterior con el Leap, sin que ello suponga un perjuicio para la robustez. También se ha pensado en el cambio del lenguaje de programación de las aplicaciones a C++, por su mayor eficiencia frente al lenguaje empleado en este trabajo, si bien no se ha apreciado un bajo rendimiento ni un framerate reducido por el hecho de haberlo implementado en AS3.

Bibliografía

[1] Leap Official Site - https://www.leapmotion.com

[2] Leap Wikipedia - http://en.wikipedia.org/wiki/Leap_Motion

[3] Leap Motion Light Pattern - http://www.youtube.com/watch?v=UI5EBzU_QqM

[4] Leap Motion Teardown - http://www.youtube.com/watch?v=uF0NSUmxFYA

[5] OpenLeap - http://www.youtube.com/watch?v=QQMGvWaFhuo

[6] Leap AS3 API - https://logotype.se/leapmotion/docs

[7] Away 3D API - http://away3d.com/livedocs/away3d/4.0

[8] HP Envy 17 (Leap integrado) - http://www.pcworld.com/article/2049000/leap-motion-comes-to-the-pc-in-hp-envy-17-notebook.html

[9] Leap Motion en los hospitales - http://wwwhatsnew.com/2013/10/15/tedcas-y-leap-motion-permiten-llevar-minority-report-a-los-hospitales/

[10] Teardown of Leap - http://techon.nikkeibp.co.jp/english/NEWS_EN/20130909/302061/ (1 y 2)

[11] ¿Qué tecnología utiliza el Leap? - http://www.extremetech.com/extreme/131159-leap-motion-will-it-make-you-a-magician-or-is-it-just-handwaving/ (1 y 2)

[12] Airspace – https://airspace.leapmotion.com/

[13] Blog oficial - http://blog.leapmotion.com/

[14] Leap Motion para Chrome - https://chrome.google.com/webstore/detail/dextype-for-leap-motion/nmdipjbemendfkmjpebpokbbmghdligc

[15] Leap Motion Desktop Windows 8 - http://www.youtube.com/watch?v=21LtA5-wiwU

[16] Inside the Smart Home – Richard Harper, ed. Springer-Verlag London Limited, 2003

[17] Robust Hand Gesture Reconigtion Based on Finger-Earth Mover's Distance with a Commodity Depth Camera – Zhou Ren, Junsong Yuan, Zhengyou Zhang. MM'11 Proceedings of the 19th ACM international conference on Multimedia, pp. 1093-1096

[18] Analysis of the Accuracy and Robustness of the Leap Motion Controller - Frank Weichert , Daniel Bachmann, Bartholomäus Rudak, Denis Fisseler. Sensors 2013, 13, 6380-6393

[19] Human-Computer Interaction Based on Hand Gestures Using RGB-D Sensors - José Manuel Palacios, Carlos Sagüés, Eduardo Montijano, Sergio Llorente. Sensors 2013, 13, 11842-11860.

[20] 3D Imaging for Hand Gesture Recognition: Exploring the Software-Hardware Interaction of Current Technologies – Frol Periverzov, Horea T. Ilies. 3D Research, 03, 2012.

[21] Gesture Recognition and Control Part 1 – Kamal K. Vyas et al. International Journal on Recent and Innovation Trends in Computing and Communication, Vol 1, Issue 7, 2013, pp. 575-581.

[22] Time-of-Flight Sensors in Computer Graphics – Andreas Kolb, Erhardt Barth et al. Eurographics 2009.

[23] Asus Xtion Pro Live - http://www.asus.com/Multimedia/Xtion_PRO_LIVE/

[24] Kinect - http://msdn.microsoft.com/en-us/library/jj131033.aspx

[25] Camboard Nano - http://www.pmdtec.com/products_services/reference_design.php

[26] Infinneon IRS10x0C - http://www.infineon.com/dgdl/3D+Image+Sensor+IRS10x0C_PB_final_v2.pdf?folderId=db3a3043191a246301192dd3ee2c2ae4&fileId=db3a30433e82b1cf013e847e27e703e9

[27] R.Y. Wang and J. Popovic, Real-time hand-tracking with a color glove. ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH 2009, article 63.

[28] Google Earth con Leap - http://blog.leapmotion.com/post/55639018480/leapinto-google-earth

[29] Segment-Based Stereo Matching Using Belief Propagation and a Self-Adapting Dissimilarity Measure - Andreas Klaus, Mario Sormann and Konrad Karner, ICPR '06 Proceedings of the 18th International Conference on Pattern Recognition – Volume 03, pages 15-18

[30] A comparative survey on invisible structured light – David Fodi, Tadeusz Sliwa and Yvon Voisin, Proc. SPIE 5303, Machine Vision Applications in Industrial Inspection XII, 90 (May 2004)